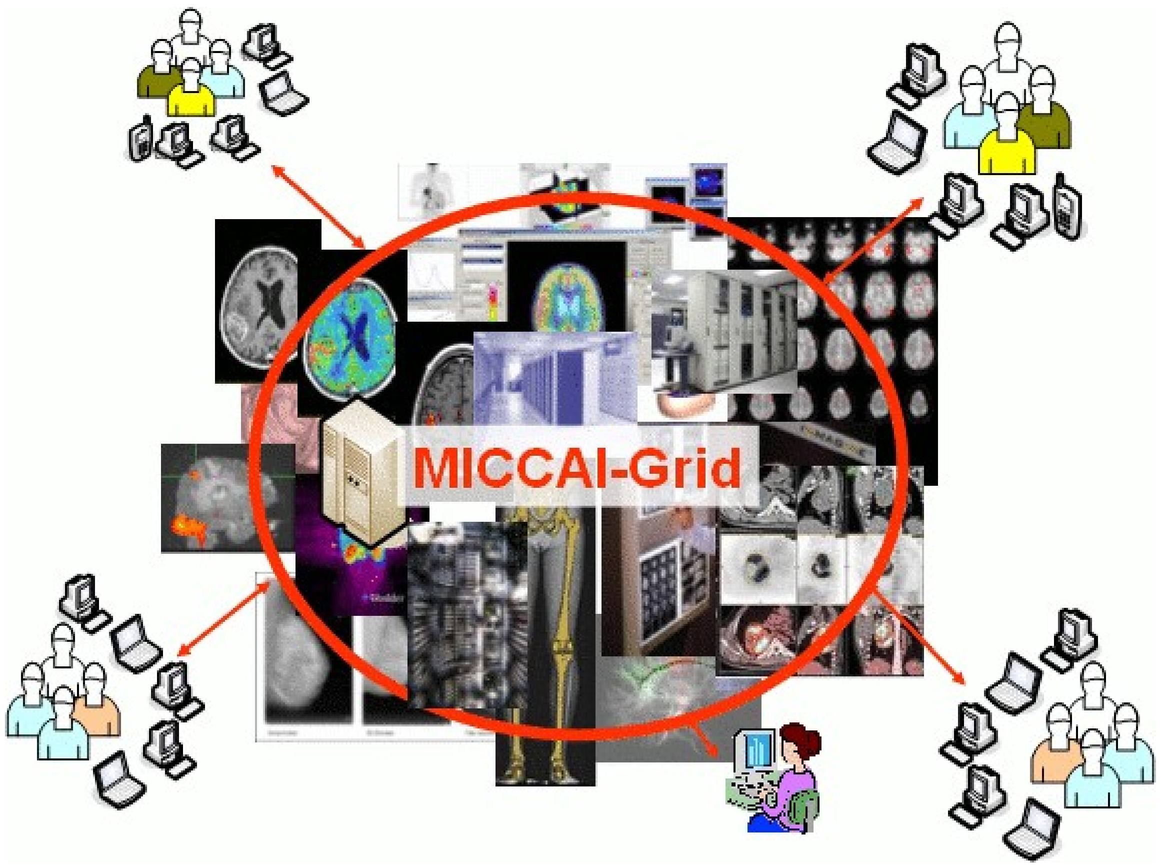
MICCAI-Grid Workshop

Medical imaging on grids: achievements and perspectives

New-York (NY), September 6, 2008.



Silvia D. Olabarriaga

Diane Lingrand

Johan Montagnat

(Editors)

http://www.i3s.unice.fr/~johan/MICCAI-Grid

Preface

Medical image computing raises new challenges related to the scale and complexity of the required analysis, for example in studies that require the federation of large data sets or in complex models and processing. Grid technology are addressing problems related to large data sets manipulation over wide computing networks, providing tools for exchanging data and computing power and additionally serving as a vector for structuring the user communities as they enable cross-enterprises collaborations. In the medical imaging area, grids provide a foundational layer that can be exploited *e.g.*, to build patient-specific models, reduce computing time to meet clinical practice constraints, algorithms validation and optimization, or collaborative studies on rare diseases. Specific grids initiatives are emerging worldwide, demonstrating a growing interest from the health community for such infrastructures and impacting the way to conduct medical research. However, deploying medical image analysis applications on grid infrastructures requires a proper understanding of the specific needs in this area.

The Workshop on *Medical imaging on grids: achievements and perspectives* (MICCAI-Grid) was organized with the goals of providing an opportunity to demonstrate the current achievements of grid technologies within the medical imaging community; identifying the fundamental problems limiting the adoption of existing systems and methods; and stimulating the community to build new collaborations by taking advantage of the sharing capabilities of grids. The workshop was organized in conjunction with the MICCAI conference in New York, USA, in 2008. Seventeen papers were submitted to the workshop, representing research developed in 10 countries around the globe. Each paper has been reviewed by three independent reviewers from an international scientific committee with representatives from Europe, North and South America, and Asia.

The final program consists of seven papers selected for full presentations, three papers selected for short presentations, and additionally two invited talks. The invited talks by Stephan G. Erberich and David Manset illustrate successful stories from projects in the USA and Europe that are applying grid technologies for medical imaging applications, namely *Globus MEDICUS, Health-e-Child* and *neuGRID*. The papers discuss a large variety of topics, including frameworks for sharing imaging data and algorithms [Tromans et al. (page 43)], workflows for medical image analysis [Glatard et al. (page 33), Krefting et al. (page 23)], and aids to run and manage medical imaging software on grids [Ruiz et al. (page 13), Acher et al. (page 95)]. The presented work additionally illustrates how the large computing capacity of grids have enabled medical imaging studies that would not have been possible to accomplish otherwise [Pernod et al. (page 55), Aoun et al. (page 75), Soleman et al. (page 65)]. Finally, the papers also report about the deployment of grid systems in clinical environments [Niinimaki et al. (page 3)] and explore the potential of grid-enabled simulators of imaging devices for virtual radiology [Camarasu et al. (page 85)].

This interesting program represents the end of a journey that was possible due to the collaboration of many (organizers, reviewers, invited speakers, authors, sponsors). The MICCAI 2008 organization provided shelter to this workshop in this important conference. The members of the program committee promptly and enthusiastically embraced the idea, prepared high-quality reviews of the papers and (!) returned them sharp on time. The invited speakers were willing to leave their busy routines and share their expertise at the workshop in NY. The authors trusted the workshop, submitted their research work, sending materials in the indicated format, within the page limit, at the requested time (!). And last but not least, the workshop sponsors (Virtual Laboratory for e-Sciences project, NL and GDR ASR, CNRS, FR) provided funds to facilitate the trip of the invited speakers.

We wish the MICCAI-Grid workshop will leave up to this promising start.

New York, 6 September 2008

Johan Montagnat and Diane Lingrand
I3S Laboratory
CNRS / Univ. Nice-Sophia Antipolis
France

Sílvia D. Olabarriaga
Academic Medical Center
University of Amsterdam
The Netherlands

List of reviewers

- Christian Barillot, CNRS / IRISA, FR
- Ignacio Blanquer Espert, University Politecnica de Valencia, ES
- Bob van Dijk, Free University, Amsterdam, NL
- Stephan Erberich, Univeristy of South California, US
- Alejandro Frangi, Pompeu Fabra University, ES
- Marco Antonio Gutierrez, Heart Institute, Sao Paulo, BR
- Sennay Ghebreab, University of Amsterdam, NL
- Tristan Glatard, University of Amsterdam, NL
- Ron Kikinis, Harvard Medical School, US
- Diane Lingrand, University Nice - Sophia Antipolis / IS3, FR
- Isabelle Magnin, INSERM-CNRS / Creatis-LRMN, FR
- David Manset, Maat-G, FR
- Johan Montagnat, CNRS / I3S, FR
- Richard McClatchey, University of West England, UK
- Toshiharu Nakai, National Center for Geriatrics and Gerontology, JP
- Sílvia D. Olabarriaga, University of Amsterdam, NL
- Xavier Pennec, INRIA Sophia Antipolis, FR
- Alfredo Tirado Ramos, University of Amsterdam, NL
- Omer F. Rana, Cardiff University, UK
- Daniel Rueckert, Imperial College, UK
- Frank J. Seinstra, Free University, Amsterdam, NL
- Arthur Toga, University of California, Los Angeles, US

List of papers

Invited talk: Stephan G. Erberich, PhD, Director Center for Health Informatics (CHI), University of Southern California (USC) *Globus MEDICUS - A HealthGrid Platform for Diagnostic and Therapeutic Medical Image Exchange in Clinical Research and Healthcare* 1

Building a Community Grid for Medical Image Analysis inside a Hospital, a Case Study, Marko Niinimaki, Xin Zhou, Adrien Depeursinge, Antoine Geissbuhler, and Henning Muller. 3

Simplifying the Utilization of Grid Computation using Grid Wizard Enterprise, Marco Ruiz, Neil C. Jones and Jeffrey S. Grethe. 13

Simplified Grid Implementation of Medical Image Processing Algorithms using a Workflow Management System, Dagmar Krefting, Michal Vossberg and Thomas Tolxdorff. 23

From gridified scripts to workflows: the FSL Feat case, Tristan Glatard and Silvia D. Olabarriaga. 33

The Application of a Service-Oriented Infrastructure to Support Medical Research in Mammography, Christopher Tromans, Sir Michael Brady, David Power, Mark Slaymaker, Douglas Russell and Andrew Simpson. 43

Invited talk: David Manset, PhD, Director of Biomedical Applications - maatGknowledge, ES, France. *Healthgrids, From Revolution to Evolution - Experiences from the Health-e-Child and neuGRID European Projects* 53

Multiple Sclerosis Brain MRI Segmentation Workflow deployment on the EGEE Grid, Erik Pernod, Jean-Christophe Souplet, Javier Rojas Balderrama, Diane Lingrand and Xavier Pennec. 55

Large scale fMRI parameter study on a production grid, Remi S. Soleman, Tristan Glatard, Dick J. Veltman, Aart J. Nederveen, Silvia D. Olabarriaga. 65

Validation of the Small Animal Biospace Gamma Imager Model Using GATE Monte Carlo Simulations on the Grid", Joe Aoun, Vincent Breton, Laurent Desbat, Bruno Bzeznik, Mehdi Leabad and Julien Dimastromatteo. 75

Towards a Virtual Radiological Platform Based on a Grid Infrastructure, Sorina Camarasu, Hugues Benoit-Cattin, Laurent Guigues, Patrick Clarysse, Olivier Bernard, Denis Friboulet. 85

Issues in Managing Variability of Medical Imaging Grid Services, Mathieu Acher, Philippe Collet and Philippe Lahire. 95

Globus MEDICUS - A HealthGrid Platform for Diagnostic and Therapeutic Medical Image Exchange in Clinical Research and Healthcare

Stephan Erberich

Director Center for Health Informatics (CHI), University of Southern California (USC)
erberich@usc.com

Invited talk

Summary

Medical image management remains one of the most challenging fields in health informatics. Both clinical and research image workflows require some form of publication, storage, and discovery. Data scale and appropriate semantic description /annotation of images create significant complexity of these workflows. On the other side expectations of the user community are demanding distributed access and reliable uninterrupted workflows reusing existing applications and infrastructure. In addition regulatory compliance, e.g. HIPAA and FDA conformance, are required specifically for clinical and research use.

Radiological imaging presents the advantage that standards for storage and communication of images exit and thus present an ideal candidate for middleware solutions. The Digital Imaging and Communications in Medicine (DICOM) standard (www.nema.org) defines the image and meta-data format and the network protocol between medical imaging devices. Since then DICOM has become the de-facto standard for medical imaging adopted by all medical equipment vendors.

The latest DICOM standard, release v3, defines the image format and a peer-to-peer network transport protocol. DICOM objects contain meta information about patient, study, series, and image attributes. Together image format, meta data, and network model, define a medical image specific application domain, referred here as DICOM domain.

Healthcare image management comprises primarily (i) storage, (ii) availability (fault-tolerance and disaster recovery), and (iii) recently interoperability and information exchange. However DICOM does not provide image management, but leaves it to the vendors to implement. As a consequence, we see a variety of silo implementations today making it almost impossible to achieve interoperability. The Integrating the Healthcare Enterprise (IHE, www.ihe.net) initiative has be created to address the issue of interoperability with the approach to interface existing silos with significant success in recent years. The latest interface, the cross-enterprise document exchange (XDS) and its medical imaging cousin XDS-I are taking shape.

However XDS and XDS-I only address interoperability. Still missing is an open standards based architecture that provides an overarching and scalable infrastructure to implement all aspects of healthcare image management (i-iii). Such an infrastructure must keep the integrity of DICOM intact to preserve compatibility to existing and future investments into imaging devices, Picture Archiving and Communication Systems

(PACS), and display workstations a $1.9 Billion annual business.

Globus Alliance Grid technology (Globus Toolkit, www.globus.org) is an open Grid architecture providing reliable industry standards for the most challenging problems associated with network collaborative environments: (i) high-speed reliable data transport utilizing high-bandwidth networks (cite), (ii) enterprise level security (data, authentication, authorization), (iii) large scale data management and replication, and (iv) publication, discovery, sharing, and use of distributed, independently owned and operated computational, storage, and data resources federated in the Grid as web-services (WS, www.oasis-open.org). The Grid paradigm spawns a virtual organization (VO) over public and/or private networks between resource providers, e.g. imaging devices, and consumers, e.g. radiologists or researchers.

The Globus MEDICUS project (Medical Imaging and Computing for Unified Information Sharing) is a seamless integration of DICOM devices into Grids. As such MEDICUS builds on the overarching Grid infrastructure to implement (i-iii). In the Grid, medical images become transparently available anywhere within a VO, comparable to a Regional Health Information Organizations (RHIO) of hospitals or practices, and between VOs.

Because the images are available within the Grid infrastructure, standards based WS, e.g. storage, image processing, data mining, become easier to develop and to deploy. We believe that such an overarching open standards infrastructure, like MEDICUS, will enable image availability at the point-of-care and thus helps to eliminate redundant imaging to the benefit of the patient and subsequently reduces cost.

Two image workflow use-cases will be described using the Globus MEDICUS platform: (i) Patient Centric Authorization clinical workflow with County, State, Federal PKI credentialing and (ii) Multi-Center Clinical Trial image workflows in the Childrens Oncology Group (COG) 40 sites production Grid.

Building a Community Grid for Medical Image Analysis inside a Hospital, a Case Study

Marko Niinimaki[1], Xin Zhou[1], Adrien Depeursinge[1], Antoine Geissbuhler[1], and Henning Müller[1],[2]

[1]Medical Informatics Service, Geneva University Hospitals and University of Geneva, Geneva, Switzerland
[2]Business Information System, University of Applied Sciences Sierre, Switzerland

Abstract

This paper describes steps taken to create an internal Grid network to support medical image analysis at the Geneva University Hospitals. While computing resources today are inexpensive, the real bottleneck of their usage in many institutes is the difficulty to harness them for a particular application and to maintain an infrastructure. To overcome this problem we have created a community Grid that provides the researchers a simple interface to a computing cluster inside the hospitals. The fact that no data has to leave the institution for the computation makes the access to the resources easier for researchers and removes some of the security constraints. Reusing existing computers on the hospital network, an automatic setup to distribute the software, using conceptualization techniques, and using a robust Grid solution based on Linux and ARC (Advanced Resource Connector) all limit the required maintenance and can quickly give researchers access to computing power.

We describe the components of our setup, its usage with a real application, and demonstrate its performance.

Contents

1 Introduction 3

2 Challenges for creating a computing infrastructure inside a hospital 5

3 Solution guidelines 6

4 An application 8

5 Conclusions 10

1 Introduction

Modern hospitals produce enormous amounts of data in all departments, from images, to lab results, medication use, and release letters. Since several years these data are most often produced in digital form, making them accessible for researchers to optimize the outcome of the care process and analyze all the available data across patients. The Geneva University Hospitals (HUG) are no exception to this with its daily radiology department's output of over 70'000 images in 2007, majority of them tomographic slices, sometimes in temporal series with a relatively small size (around 512 kB). Other departments such as cardiology, dermatology, or pathology, likewise produce large amounts of visual data that need to be stored and could be analyzed for diagnostic aid tools [4].

Content–based medical image retrieval in general has the goal to allow for retrieval of similar images or cases over very heterogeneous image collections to help the diagnostic process [15, 20]. For large collections, this is a time consuming process, even if for single images feature extraction is often fast [21]. Parallelization of the processing is needed to keep up with the flood of images.

Different technologies can be applied to parallization. In [14], we utilized the Parallel Virtual Machine software tool to embed in the program code how to distribute the processing in a few (locally connected) computers. Cluster computing systems, or Local Resource Management Systems (LRMSes), like Condor [11] provide a higher level command language in order to, typically, send a program and its input to be run in any computer of the cluster. Grid technologies enable us to connect several clusters, even if they run different kinds of LRMSes.

With their security infrastructure, standardized job execution and data access interfaces, Grid technologies provide tools for complicated distributed tasks. "Gridified" image retrieval has been proposed several times, most often using external clusters made available by large–scale research projects in the domain [12, 16]. Sustainability of these solutions and a business model still need to be developed.

Most approaches for image retrieval on Grids concentrate on the visual feature extraction as this is generally the most time–consuming part. Projects such as caBIG[1] also allow for image retrieval in Grid environments but rather relate to text–based retrieval, which is less computationally expensive.

Previously, the medGIFT[2] group at the Medical Informatics Service at the HUG successfully gridified their GNU Image Finding Tool (GIFT[3])–based medical image retrieval application [17, 8]. However, the actual usage of this gridified version was limited to a single quad–core server, or external Grid resources, as described in [21], made available by the KnowARC[4] EU research project.

While Grids in general can be characterized as resource sharing in multi–institutional organizations [7], Grid technologies can be applied to access resources inside just one organization, as in our case. The benefits of this are (1) conformance with the controlled and strict access policies needed in organizations that handle confidential data, (2) the possibility to connect resources at different sites within the organization — for instance in different subnetworks in buildings that can be many kilometers apart, and (3) the possibility of later expanding the analysis to an inter–organizational level while using the same interfaces.

Like most hospitals, the HUG do not have any central computing infrastructure at the moment that would allow researchers to run computationally intensive applications, for instance in a cluster of dedicated computers or a mainframe computer. However, over 6'000 desktop computers are available on the network of the hospital for computation, and another 5'000 on the University of Geneva network. Even a partial use of these resources, for example at night or on a voluntary basis, could help to fulfill the most urgent computational needs for medical image analysis that would allow much more advanced algorithms. Currently, most of these computers are mainly used to access patient records from central databases, an activity that consumes only very little of the resources of the computers. At the same time, care must be taken that the

[1]http//cabig.nci.nih.gov/
[2]http://www.sim.hcuge.ch/medgift/
[3]http://www.gnu.org/software/gift/
[4]http://www.knowarc.eu/

analysis tasks do not exhaust the hospital network resources.

In this paper we demonstrate that the creation of a computing Grid within a hospital organization (such as ours) is possible, despite the many technical and particularly political concerns. We have built a test bed consisting of standard hospital desktop PCs running a LRMS (Local Resource Management System) that is capable of accepting jobs from the ARC (Advanced Resource Connector) Grid middleware [5]. The LRMS runs inside virtual machines, so that the software needed for distributed computing tasks can be run in an isolated environment. Our test bed setup was designed so that it can be controlled by the hospital IT administrators and requires minimum intervention from the part of the researchers. Existing infrastructures for software distribution as well as security policies were all taken into account.

The rest of the paper is organized as follows. In Section 2 we describe the challenges met for harnessing the desktop computing resources of a large hospital with a sometimes complicated security and political structure. In Section 3 we present our solution based on the given constraints. In Section 4, we discuss our GIFT–based application and demonstrate its performance in our internal hospital Grid. Conclusions are drawn in Section 5.

2 Challenges for creating a computing infrastructure inside a hospital

Several problems for creating Grid networks inside medical institutions were already known before the beginning of the project [13, 17]. These concern mainly the little flexibility of a large institution towards research projects that are clearly not the main objectives of the teams managing the infrastructure. This is quite understandable since the smooth running of the hospital network, networked appliances and applications is certainly a priority in an organization where people's lives can depend on them. Data security and confidentiality are similarly important and our applications can not compromise them.

On the other hand, Grid software is often relatively demanding to its environment, often requiring e.g. computers with static IP (Internet Protocol) addresses and access to specific TCP (Transmission Control Protocol) ports. While these are not difficult to be provided within hospital local area networks (LANs), they may still contradict hospital policies. Grid and LRMS systems tend to be fairly intrusive as well, requiring dedicated computing resources, whereas we would favor a setup where computing can use only a part of the resources of a normal user's desktop PC.

Thus the main problems are related to infrastructure (I) and Grid software (G). Political problems (P) form an additional factor. We can summarize these three categories and the requirements that they pose to our solution as follows:

I1 All applications within the hospital are behind a rigid firewall that does not allow us to have direct connections outside of the hospitals: all TCP ports other than 80 (http - Hypertext Transfer Protocol) and 443 (https - Hypertext Transfer Protocol over Secure Socket Layer) are blocked, and access to http/https is only possible through a proxy. The only way for us to get access to the University network was via a VPN (Virtual Private Network). Requirement: our Grid setup should not rely on external resources.

I2 Research applications such as GIFT (GNU Image Finding Tool) and also Grid middlewares are often developed under Linux, whereas Linux was not allowed to be installed on the hospital desktops at the beginning of our project. Requirement: do not assume a Linux environment.

G1 For an inexperienced programmer and system administrator, Grid systems are hard to install, maintain, and interface. Grid certificates can be difficult to obtain. Requirement: allow sufficient time for learning, use external expertise, use a temporary certificate authority, if needed.

P1 Grids are relatively unknown in hospital network departments. The lack of knowledge creates a rejection of technology for the responsible persons as they are afraid to not be able to control the infrastructure. Requirement: involve hospital IT, create a test bed for such new technology.

P2 The fragmentation of the structure of the hospital into very independent entities results often in very strict decision hierarchies and rather a tendency to reject technology and remain with an existing, restrictive structure; it is sometimes hard to find the responsible person for a particular decision and often response times for requests can be long due to the fragmentation. Requirement: be patient. Try to employ a solution that does not need much of the IT personnel's involvement.

3 Solution guidelines

Despite the problems, we were able to establish a test bed and based on it demonstrate the benefits of Grids in the domain of the hospital. A wider distribution inside the hospital is planned to be deployed for research projects in the future.

Briefly, our solution can be described as running a computing node inside Virtual Machines that, in turn, are run in desktop computers. Job submission to these computing nodes is managed by the ARC middleware. Currently, users who want to submit jobs, log in to a Linux computer that has the both the ARC server and its job submission interface. More computers can be added later both for job submission and running ARC, removing the "single point of failure". Virtualization was chosen as the processes running in the virtual machines can be in another operating system than the host (in our case Linux) and the processes running in a virtual machine have no access to potentially confidential files on the host operating system.[5]

The requirements were considered in the solution as follows:

I1 A local Grid information system was employed inside the hospital network to control the information of the cluster. No external connectivity is required. However, the solution can support job submission to both internal and external resources, when available. The internal computing resources do not need any external connectivity.

I2 The desktops run the hospital standard issue Windows XP, completely controlled by the hospital IT (by standardized distributed installation images). A specific distribution package was created for these computers, containing VMPlayer[6] and the Virtual Machine image. The image contains a Debian Linux, with all the software packages required for image analysis, and the LRMS for receiving packets to be computed and sending back the results. The Linux image was to be kept as small as possible to limit its influence on the network traffic when installing the images on a larger number of desktops.

G1 Our decision to use ARC and the Condor LRMS was based on the project personnel's expertise in these tools, and enabled us to create a working Grid and cluster setup quickly. Our own certificate authority (CA) was used when getting certificates from official Grid CA's took too much time.

P1 Before our Grid system could potentially be deployed in the hospital at a larger scale, a smaller set of desktop PCs was assigned for this purpose. The network administrators wanted to assure thorough testing before any larger–scale use. For our test bed, 20 desktop PCs of the type Compaq Evo 510

[5]These are often cited as benefits of a Virtual Machine Grid, see e.g. [6]. For an overview of virtualization techniques see [1].
[6]`http://www.vmware.com/`

(Pentium 4, 2.8 GHz) were made available to us.[7] A standard hospital installation of Windows XP with antivirus software was used as the operating system. The network connectivity of these computers was, likewise, the same as for any hospital desktop, i.e. they belong to the same sub network as other desktops in the main hospital area, and gain their IP addresses from the same address pool.

P2 Originally, we wanted to run Linux in the Virtual Machine images with Network Address Translation (NAT); they were able to initiate connections to the network through the host computer, but incoming connections would be impossible. Therefore, the Virtual Machine images would not have needed to have site–wide IP addresses managed by the hospital IT.

The LRMS, in our case Condor, is run inside the virtual machines and communicates with the Manager node. In a setup called Bridged Model, virtual machines can have IP addresses that are accessible from the other nodes. However, this means that the Virtual machines need to gain their IP addresses from the hospital's DHCP (Dynamic Host Configuration Protocol) server, and that is strongly discouraged for administrative reasons. Some workarounds exist, including a VPN (Virtual Private Network) tunnel from the execution nodes to the manager, and Condor GCB (Generic Connection Broker) [19]. We originally relied on GCB in the following setup. The worker nodes were configured to establish connections to a computer running GCB (actually this is the same computer as our Condor Manager but this is not important in this context). GCB then forwards the connections to Condor Manager, letting the execution node and the Manager communicate as if there was no NAT. However, this approach did not prove to be reliable enough. If the connection was dropped due to network lags or other minor problems, it failed to re–establish. Our current model forces a MAC (Medium Access Control) address to the Virtual Machine at startup, and the DHCP (Dynamic Host Configuration Protocol) administrator has allocated IP addresses for these specific MACs. However, this adds unnecessary administration overhead to the system and another solution is envisioned for the future to allow for a better scalability.

In practice, the setup of our environment is shown in Figure 1. One local ARC server is currently in production. The same computer acts as a manager node for the Condor LRMS. By installing more ARC/Condor servers in the hospital, we will supply redundancy so that the system will work even if one of the servers fails. A Grid Index Information Service (GIIS) contains the details of the ARC servers, including number of active nodes, and the runtime environments (like Java) provided by these servers.

A grid job description contains the input images for further feature extraction, the GIFT system as source code, and a small program to compile and run GIFT with the images. ARC's GridManager receives the requests and pushes them to Condor. ARC then supervises their execution and receives their output. The output is stored at the ARC server until the user issues a download command.

Condor execution nodes are run in Virtual Machines (VMs), as shown on the left side of the figure. The Linux system inside the VM contains a C compiler and other tools needed to compile and execute GIFT feature extraction.

Since the execution nodes are run in desktop PC's, the system must be resistant to failures – after all, PC users regularly re–boot their computers. Here, we have used a job manager called GridJM[8]. In the case of failure, the job manager simply re–submits the job until it succeeds or a re–submit limit has been reached.

Grid users log in to the same Linux computers where we run the ARC servers, i.e. job submission is done in those computers, too. This is not an optimal solution, but needed since the user's in general have only Windows desktop computers and job submission is easier using Linux.

[7]It should be noticed that the processor of these computers does not support virtualization extensions thus making the virtual machines run slower than more recent hardware; for details about hardware level virtualization support see e.g. [9].

[8]http://www.tcs.hut.fi/ aehyvari/gridjm/

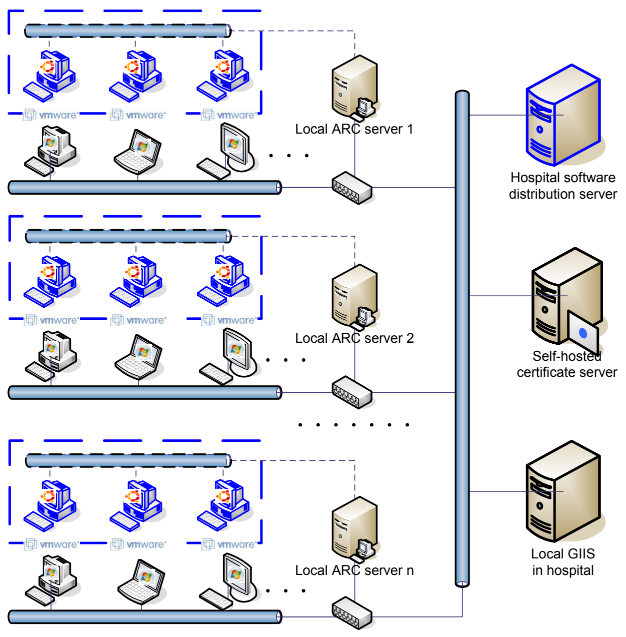


Figure 1: The main components of our setup.

Finally, the hospital IT administration controls the Windows operating system environment completely. The VMware installation and Linux VMware images are automatically installed in the computers using an automated installation system provided by the hospital.

4 An application

Our setup was created for the needs of the MedGIFT group of the Medical Informatics Service and the application uses a gridified version of the GNU Image Finding Tool (GIFT) created in a research project at Geneva University several years ago. In practice this means the data from a large image database is re–packaged in chunks of N images, the packages and their processing instructions are sent to computing nodes where they are processed in parallel, and the results are recovered from the nodes. A job description language, in our case ARC's XRSL (Extended Resource Specification Language, see [18]), is used to request the computing.

In line with out previous research ([21]), we have used the ImageCLEF (see [3]) database of almost 50'000 images with a package size of 1'000 (except for the last package, naturally). The flow of a single job is as follows:

- the user submits the job to the ARC node;
- the ARC middleware pushes the job to the LRMS;
- execution in the LRMS takes place in the virtual machines; the virtual machines are waiting for jobs since they have been deployed by the hospital installation system;
- the user runs a command to receive the output of the job.

4 CPU server	709 min
37 CPU remote clusters	537 min
20 CPU local VM cluster	240 min

Table 1: Comparison of running jobs in a local 4 CPU cluster, remote clusters of the KnowARC project, and our hospital VM cluster.

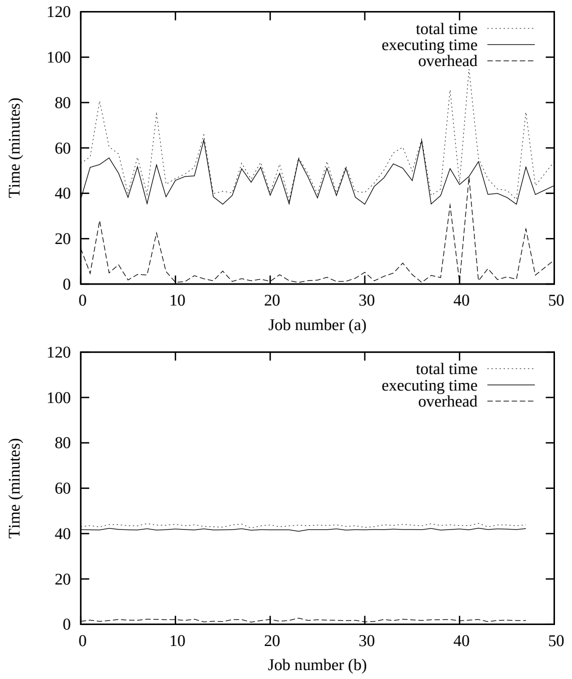


Figure 2: Comparison of the time consumption with remote (a) and local (b) infrastructures.

In the case of a single job it is important to evaluate (a) the performance of a VM–based Linux installation compared to a native one in an identical computer and (b) delays imposed by job preparation by the middleware and the LRMS, and the transfer of the package to the execution node. The exact figures (32 minutes in native Linux, 41 minutes in the virtual machine, 45 minutes per average by remote ARC resources) show that the overheads caused by the virtual machine, ARC, and the LRMS are negligible compared to actual computing time.

A more concise result is shown in Table 1. It consists of running a large set of image analysis jobs under the control of a GridJM, as describe in our previous paper [21]. In practice, a job generator creates the packages and the XRSL job description files. They are passed to GridJM that submits them to the ARC resources that are available to the user, resubmits in the case of a failure, and recovers the results.

In Table 1, we compare feature extraction in a quad–core server and a small Grid of 37 CPUs in remote locations in several countries. In all these cases, the number of image packages is 50, containing 1’000 images each (the last package slightly less), each package about 190 MB. Here, we compare the results of that paper in a situation where we have the same data sets executed in our Virtual Machine cluster.

In [21], we presented the execution with remote Grid resources, and predicted that using a local cluster will largely reduce the overall execution time. Figure 2 shows a comparison between using remote Grid resources and the local cluster, which proves this assumption. The local cluster used in this test consists of

20 desktops (five–year–old Compaq Evo desktop computers with Pentium 4–2.8 GHz, 768MB memory). However, since our package generator could not supply packages fast enough, only a maximum of 10 jobs run in parallel.

An identical Virtual Machine image was used in each of the desktops used taking 300MB of memory. From Figure 2 we can see that the time spent on execution is quite uniform ("flat lines"), as each computing node is set up in the same way. The main benefit comes from the time gained from scheduling, queuing, and transfer of data. This overhead is reduced from 10 minutes on average per package (in the case of remote resources) to only 1 minute. With only 10 desktops employed in the local cluster for testing, the total time is reduced to less than half. No jobs were required to be resubmitted, either.

The results can be improved easily by using a smaller package size, so that package generation does not become a bottleneck.

Finally, we present some technical details about the Virtual Machines. The VM "image" that contains the Linux system runnable inside the VM is distributed as a single, compressed 317 MB file. The Linux system in the Virtual Machine image is based on Debian 3.1. The principal software packages and their sizes are as follows: GNU compiler, linker and header files – ca. 30 MB; ImageMagick image conversion software – ca. 27 MB; the Java environment – ca. 170 MB; and Condor – ca. 100 MB.

The impact of running the analysis was not significant for the network, since only 50 files of size 190 MB (and small result files) were transmitted. The impact for the desktop users will be more visible: when the Virtual Machine is running, half of the PC's memory is used by it. Moreover, we measured the CPU performance using NovaBench software, first when no other software was running in the PC, and later during the analysis. The results indicate that the analysis used about 50% of the PC's CPU capacity, too.[9]

5 Conclusions

In this paper we have demonstrated the feasibility of an internal hospital Grid. We discussed the challenges imposed by strict requirements of the organization, and how to find solutions to the various problems. An application in content–based medical image retrieval using our Grid cluster is described and performance measurements are shown.

This article solves several problems that exist in many medical institutions. Data are produced in quickly increasing quantities and analyzing these data is required to improve care process. Unfortunately, very few central computing resources exist in hospitals for research projects to analyze such large amounts of data using modern, complex algorithms. Desktop computers on the other hand exist in large quantities in most medical institutions and reusing them for computation seems a good idea. Our setup using virtualization for the Grid solves several problems:

- the Linux operating system can be used for the image analysis;
- security is improved as the virtual machine does not have access to the host operating system;
- the user remains in control and can stop a virtual machine in case the processing is slowing down the computer or he needs the computing power for other tasks.

After all, our experiences with these Grid technologies are very positive. The setup of the Grid using the standard automatic software distribution system of the hospitals was very quick. Maintenance of such a

[9]Without the VM: 35 Mflops/s, 6.4 M integer operations/s, when analysis running in the VM: 18 Mflops/s, 2.3 M integer operations/s.

system is also relatively easy as a maximum of steps for setup and maintenance are automated. Maintaining the software environment in the virtual machines, too, has been easier than expected since we have been able to provide a single virtual machine image, containing all the components needed for image analysis jobs. It should be foreseen that the maintenance will become more demanding if completely different type of analysis, with different software components, will be needed in the future. However, so–called dynamic runtime environments (see [2]) for Grid middleware will facilitate such tasks.

Overhead when using external Grid resources caused by job scheduling and transmissions of large datasets are also reduced significantly. The system is thus ready for a larger deployment and the use for other research projects than only image retrieval. For the time being, we have only tested our system with tasks containing a single step (e.g. feature extraction from a set of images) running in parallel. In the future, more complicated workflows with jobs producing output for other jobs are likely to enter the scene. In order to execute these kinds of work flows in our Grid, ARC integrated with Taverna workflow manager (see [10]) will be tested.

Acknowledgements

This work was partially funded by the European Union 6th Framework program in the context of the KnowARC research project (contract number 032691). We would also like to thank the network group of the HUG for their support in the project and the quick responses we often received.

References

[1] Keith Adams and Ole Agesen. A comparison of software and hardware techniques for x86 virtualization. In *ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 2–13, New York, NY, USA, 2006. ACM. 5

[2] D. Bayer, T. Bhindi, F. Orellana, A. Waananen, B. Konya, and S Moeller. Dynamic runtime environments for grid computing. In Marian Bubak and Kazimierz Wiatr Michal Turalad and, editors, *CGW'07 – Proceedings of Cracow'07 Grid Workshop*, pages 155–162, October 2007. 5

[3] P. Clough, M. Grubinger, T. Deselaers, A. Hanbury, and H. Müller. Overview of the ImageCLEF 2006 photographic retrieval and object annotation tasks. In *Working Notes of the 2006 CLEF Workshop*, Alicante, Spain, September 2006. 4

[4] Adrien Depeursinge, Henning Müller, Asmâa Hidki, Pierre-Alexandre Poletti, Alexandra Platon, and Antoine Geissbuhler. Image–based diagnostic aid for interstitial lung disease with secondary data integration. In *SPIE Medical Imaging*, San Diego, CA, USA, February 2007. 1

[5] M. Ellert, M. Grønager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J. Langgaard Nielsen, M. Niinimäki, O. Smirnova, and A. Wäänänen. Advanced resource connector middleware for lightweight computational grids. *Future Generation computer systems*, 23(2):219–240, 2007. 1

[6] R. Figueiredo, P. Dinda, and J. Fortes. A case for grid computing on virtual machines. In *Proc. International Conference on Distributed Computing Systems (ICDCS)*, May 2003. 5

[7] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *The International Journal of Supercomputer Applications*, 15(3), Summer 2001. 1

[8] A. Hidki, A. Depeursinge, J. Iavindrasana, M. Pitkanen, X. Zhou, and H. Müller. The medgift project: perspective of a medical doctor. *Journal of Medical Imaging Technology*, 2007. 1

[9] Intel. Intel virtualization technology. *Intel Technology Journal*, 10(3), 2006. 7

[10] Hajo Krabbenhöft, Steffen Möller, and Daniel Bayer. Integrating arc grid middleware with taverna workflows. *Bioinformatics Applications Note*, 2008. 5

[11] M. Litzkov, M. Livny, and M. Mutka. Condor — a hunter of idle workstations. In *Proceedings of the 8th international conference on distributed computing*, pages 104–111, San Jose, California, USA, June 1988. 1

[12] J. Montagnat, V. Breton, and I. E. Magnin. Partitioning medical image databases for content–based queries on a grid. *International Journal of Supercomputer Applications*, 44(2):154–160, 2005. 1

[13] H. Müller, A. Garcia, J. Vallée, and A. Geissbuhler. Grid Computing at the University Hospitals of Geneva. In *Proc. of the 1st HealthGrid conference*, pages 264–276, Lyon, France, January 2003. 2

[14] H. Müller, D. McG. Squire, W. Müller, and T. Pun. Efficient access methods for content–based image retrieval with inverted files. Technical Report 99.02, Computer Vision Group, Computing Centre, University of Geneva, rue Général Dufour, 24, CH–1211 Genève, Switzerland, July 1999. 1

[15] Henning Müller, Nicolas Michoux, David Bandon, and Antoine Geissbuhler. A review of content–based image retrieval systems in medicine – clinical benefits and future directions. *International Journal of Medical Informatics*, 73:1–23, 2004. 1

[16] M. Costa Oliveira, W. Cirne, and P. M. de Azevedo Marques. Towards applying content–based image retrieval in clinical routine. *Future Generation Computer Systems*, 23:466–474, 2007. 1

[17] Mikko Pitkanen, Xin Zhou, Miika Tuisku, Tapio Niemi, Ville Ryynanen, and Henning Müller. How grids are perceived in healthcare and the public service sector. In *HealthGrid*, Chigaco, U.S.A., June 2008. 1, 2

[18] O. Smirnova. *Extended Resource Specification Language (XRSL)*. The NorduGrid Collaboration. NORDUGRID-MANUAL-4. 4

[19] Sechang Son and Miron Livny. Recovering internet symmetry in distributed computing. In *Proc. Cluster Computing and the Grid (CCGrid 2003)*, May 2003. 3

[20] Lilian H. Y. Tang, R. Hanka, and H. H. S. Ip. A review of intelligent content–based indexing and browsing of medical images. *Health Informatics Journal*, 5:40–49, 1999. 1

[21] X. Zhou, M. Pitkanen, A. Depeursinge, and H. Müller. A medical image retrieval application using grid technologies to speed up feature extraction. In *ICT4Health*, Manila, Philippines, February 2008. 1, 4, 4

Simplifying the Utilization of Grid Computation using Grid Wizard Enterprise

Marco Ruiz[1], Neil C. Jones[2] and Jeffrey S. Grethe[1]

[1]Center for Research in Biological Systems, University of California at San Diego, La Jolla, CA
[2]Computer Science Department, University of California at San Diego, La Jolla, CA

Abstract

The field of high performance computing (HPC) has provided a wide array of strategies for supplying additional computing power to the goal of reducing the total "clock time" required to complete large scale analyses. These strategies range from the development of higher performance hardware to the assembly of large networks of commodity computers. However, for the non-computational scientist wishing to utilize these services, usable software remains elusive. Here we present a software design and implementation of a tool, Grid Wizard Enterprise (GWE; http://www.gridwizardenterprise.org/), aimed at providing a solution to the particular problem of the adoption of advanced grid technologies by biomedical researchers. GWE provides an intuitive environment and tools that bridge this gulf between the researcher and current grid technologies allowing them to run inter-independent computational processes faster by brokering their execution across a virtual grid of computational resources with a minimum of user intervention. The GWE architecture has been designed in close collaboration with biomedical researchers and supports the majority of every-day tasks performed by computational scientists in the fields of computational biology and medical image analysis.

Acknowledgements: This research was supported by National Institutes of Health (NIH) grants U54-EB005149 (National Alliance for Medical Image Computing) and U24-RR019701 (Biomedical Informatics Research Network)

Contents

1 Introduction 13

2 Grid Wizard Enterprise (GWE) 15

3 Usage and Integration with External Applications 17

4 Future Work 22

1 Introduction

Research in the computational sciences seems to proceed, roughly speaking, in distinct phases. As an initial idea for a computational protocol is refined, a researcher will iteratively debug and improve applications on a small amount of handpicked data, tweaking parameters and inspecting output for correctness. Once the idea returns plausible results and can be considered publication-worthy, the program gets run on more and more data, often involving a new round of debugging as new classes of pathological inputs are discovered. During this phase, the computational protocol undergoes systematic characterization: for what parameter ranges does it return valid results? Can any immediate conclusions or further hypotheses be derived from running on publicly available data? Finally, once the computational protocol is ready to be released to the scientific community, a researcher must decide how to distribute the code.

It is a lucky coincidence that the research process falls into the already useful class of embarrassingly parallel problems, that is to say, resource-intensive computational problems that can be broken into independent subunits that can run in parallel. A researcher who faces such a problem must deal with a number of tedious issues: how to determine what work needs to be done and how it should be broken into meaningful units (workload definition); how to assign individual work units to resources (application scheduling[1,3]); how to run and monitor executables (grid execution; e.g. Globus[2]); and how to deal with system-related and program-related failures (failure detection). As an example, current application scheduling frameworks (e.g. Condor[1], SGE[3]) would appear to provide the end user with an easy to use platform to broker the execution of his processes, however, unless his processes are extremely straightforward and trivial, the end user faces a daunting challenge. Most real application scheduling requests require the resolution of issues that, even in the presence of a powerful resource manager, have to be resolved by the end user: uploading the data to be processed to the cluster (localization); submit all processes to compute nodes (queue jobs in resource managers); monitor processes execution progress (real time and querying on demand); send / receive custom alert notifications (certain interesting conditions reached such as a percentage of processes completed execution); failover and recovery from cluster and environment related problems; failover and recovery from processes related problems; gathering and compilation of processing results; uploading result data to the storage resource of your choice; cleaning up the original and result data from the cluster data storage resource.

Though each of these problems can be solved in a straightforward way, the combined solutions to all of them leads to a maintenance problem, interferes with distribution, and presents an additional barrier to a scientist trying to investigate a particular problem. Two different solutions have evolved, as a way to bridge this gap: users gathering a considerable level of technical knowledge, which takes time and effort away from their actual domain problems; and the creation, (by technically savvy users and/or IT departments), of highly customized scripts and applications tailored to specific parallelization problems. Both these avenues don't provide the broader community of biomedical researchers with the ability to easily harness the growing number of clustered computational resources available to them. It is our view that a researcher expert in a particular scientific discipline should not need to also become an expert in grid computing in order to produce an application that uses grid technology. It is also our observation that the bulk of computational scientists do not have at their disposal dedicated programmers and system administrators to plan, install, configure, and maintain customized scripts or a complex heterogeneous network of computers.

To reflect these needs we have designed a system, Grid Wizard Enterprise (GWE) that facilitates the above considerations, which we derived based partly on our own experiences performing computational science in bioinformatics and partly on observing others doing the same. It is important to note that GWE is not meant to be another grid middleware package, rather, it is meant to be a large-scale job launching and management tool that bridges the gulf between the biomedical researcher and current grid middleware by:

- Providing the researcher with the ability to easily configure the heterogeneous clustered/grid resources that they have access to.
- Allowing a researcher to easily specify large parametric computational jobs using the same general syntax as is used in the command line invocation of the analysis algorithms (e.g. see P2EL in Section 4) or through integration with community developed biomedical applications (e.g. see Slicer in Section 5).
- Managing the most common house keeping tasks required to ensure end-to-end success of a computation thereby relieving the researcher of this burden.

2 Grid Wizard Enterprise (GWE)

GWE is a distributed enterprise system (Figure 1) that was designed to be a practical solution for end users to easily and effectively parallelize and broker the execution of their inter-independent processes on clustered or grid environments they have access to. This system provides a solution for the issues previously mentioned in Section 1 and with a high degree of modularity which allows third parties to extend and customize the GWE system. In order to lower the barrier for use by the typical biomedical researcher, the only requirements the system has over the environment it would be running on are to have available java 1.5 or higher and operate over a SSH enabled network.

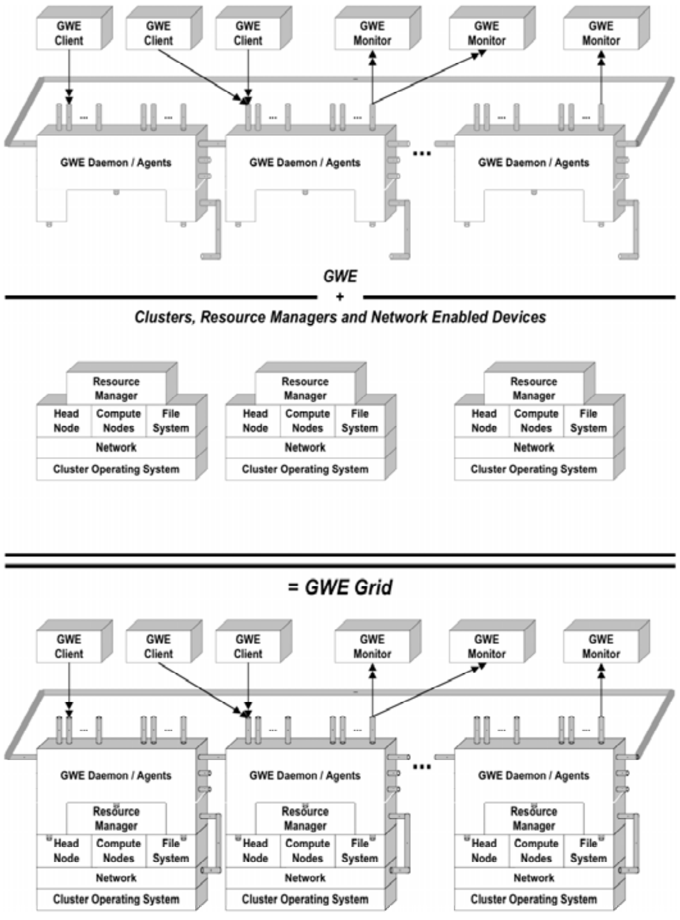


Figure 1: GWE Grid

2.1 Distributed System

From a user's perspective, GWE is composed of two subsystems. The first is the GWE client – the system running on end users machines to communicate with a 'GWE Grid' in order to query the execution status of previously submitted requests and submit new ones. This client can be access through the command line or integrated within a biomedical application.

The second is the GWE daemon – the **s**ystem(s) running on clusters' head node to serve as a listener for end user requests, as a job dispatcher and as a monitor for the respective cluster. In addition, GWE daemons can communicate with one another when a user has requested a computation to be executed on multiple clusters. Typically, end users would connect to a particular 'GWE daemon' (running on a host on a reachable TCP/IP based network) using a 'GWE client'. A GWE client's configuration consists of: the list of clusters that compose the user's accessible grid resources, SSH authentication information of all the networked resources to be accessed (clusters, file systems, etc) and locations of applications to auto-deploy.

GWE daemons are easily deployed through an automated process in the GWE client distribution and can be installed by any user with a valid SSH account on a cluster head node. At runtime, GWE daemons will silently spawn low level 'agent type' sub-systems, which are scheduled to run on the compute nodes (one 'agent' per allocated compute node) by the local cluster's resource manager. These 'agents' will be in charge of the actual execution of the processes, monitoring status/progress/results and reporting back to the respective "daemon". A GWE daemons' configuration consists of multiple behavioral parameters. Among the most important are the ones used for its 'compute node allocation policy', such as queue size (maximum number of compute nodes allocated at any given time), "hijack" timeout (maximum time a compute node can remain allocated with active jobs before "releasing" its controlling agent) and "idle" timeout (maximum time a compute node can remain allocated with nothing to do before "releasing" its controlling agent).

2.2 Architecture & Design

All GWE subsystems have been architected as a set of independent modules and frameworks, glued declaratively using the ‘Spring’ application framework. Such modules have been designed with a robust and scalable infrastructure and with multiple levels of abstraction to provide a high degree of extensibility.

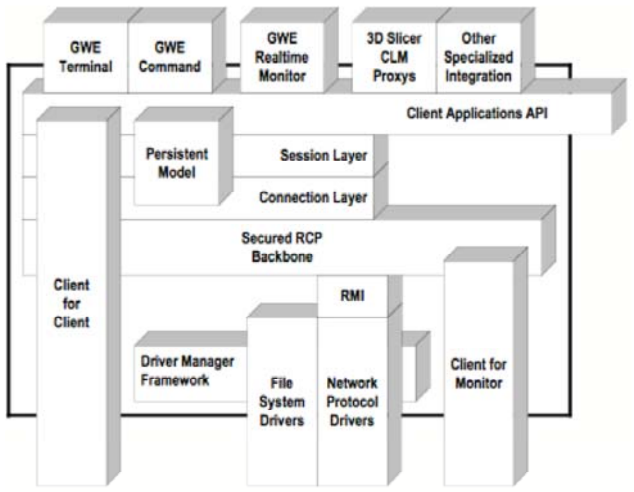


Figure 2: GWE Client Architecture

One of the most common extensible modules is the “GWE Client API” (Figure 2); used to build GWE client applications or empower applications with GWE client capabilities. Currently there are a few GWE client applications built using this API and they will be reviewed in more detail later in this paper. Internally this API contains many extensible sub-modules most of which have been further developed to reduce the effort required to add/change functionality. An example of this is the “abstract job descriptor” component; which can be extended to support languages other than P2EL, or even workflows with inter-dependent jobs.

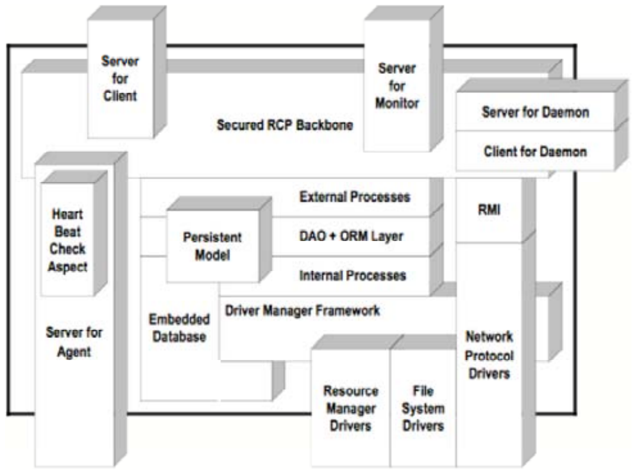


Figure 3: GWE Daemon Architecture

Finally, one set of extensible modules, which are a part of both the client and daemon (Figure 3), that deserve a special mention are the ‘grid related resource drivers’. These drivers provide GWE with means to support new types of file systems, network protocols and cluster resource managers (which are auto detected per cluster when a GWE daemon is deployed). GWE comes out of the box with drivers to support the following ‘resources’: File Systems such as Local, HTTP and SFTP; Network Protocols such as Local and SSH; and Resource Managers such as Condor, SGE and PBS. GWE daemons include other robust enterprise level features such as highly multithreaded services to avoid wait cycles and achieve maximum performance; embedded database to persist operational data (users, orders, jobs, clusters, etc) and automatic failover and recovery from cluster, environment and process specific related problems.

However, the GWE system was designed to allow any biomedical researcher to easily access their available compute resources with tools they normally utilize: the command line, SSH and domain specific applications. Therefore, the GWE inter subsystem communications infrastructure has been architected as a secured RPC backbone using a Java RMI network tunneled over SSH. This tunneling infrastructure consists of a framework, which transparently injects a series of hooks (socket proxy, heartbeat emitters and heartbeat checkers) into the communications using interceptors and AOP (aspect oriented programming).

3 Usage and Integration with External Applications

The following section details GWE based solutions for the testing of a medical imaging algorithm through GWE. It is a common scenario for researchers to fine-tune an algorithm by running it over the same data set with different values for its algorithmic parameters (large scale parameter exploration). It is also a common scenario for researchers to have large set of files they want to process using the same behavioral parameters. The example we are going to explore in this section is a combination of both cases.

3.1 Setup

All GWE clients described below have the "GWE Client API Module" bundle installed (or pre-installed) under a specific path (by simply unbundling it there), and the GWE_HOME environmental variable set to such specific path. Under "$GWE_HOME/conf", one finds GWE's configuration information; which is customized to fit the computing environment:

- **gwe-grid.xml**. Contains information about the clusters to be used (head node addresses mainly).
- **gwe-auth.xml**. Contain the user's authentication information to access the networked resources including cluster head nodes, remote file systems, etc.
- **gwe-apps.xml**. Contain information about the applications the user wants pre-installed in clusters before executing his/her orders (bundle locations, bundle architecture, reference id, etc.)

3.2 P2EL (Process Parallelization Execution Language)

In order to provide the semantics for users to easily and effectively describe a group of process invocations including all related parallelization meta-instructions, a simple but powerful language (P2EL) has been designed for GWE (and an appropriate interpreter built into it). This language is a combination of pseudo bash and pseudo VLT (Velocity Template Language). This language has semantics to define:

- **Process Invocation Templates** are Bash like, meta-templates containing iteration variables references (substitution expressions). These templates will be used to generate all the process invocations of the respective P2EL statement.
- **Substitution Expressions** are "bash like" variable expressions embedded in the template to be replaced by the corresponding value-space.
- **Iteration Variables** are variables associated with a value-space set, which when applied to a process invocation template, generate a collection of processes invocations. This construct gets its values explicitly or implicitly through numerous ways including runtime value resolving functions.
- **System Variables** are variables associated with a system and/or contextual property resolved at runtime for a specific job (e.g. the iteration number generated by GWE and user home).
- **Staging Files Instructions and Locations** instruct how to stage remote files into the context of process invocation (before executing it) and how to stage files produced by the process invocation out to selected destination locations.

3.2.1 Medical Imaging Algorithm Example

Problem

Slicer's "BSpline Deformable Registration" is a medical imaging algorithm; which resamples a "moving" image using a "fixed" one and a set of algorithmic parameters. The following is an example of this command submitted from a regular OS shell for a single invocation:

```
Slicer3 --launch /usr/Slicer3/lib/Slicer3/Plugins/BSplineDeformableRegistration
--iterations 10 --gridSize 5 --histogrambins 20 --spatialsamples 500
--maximumDeformation 1 --default 0
```

```
--resampledmovingfilename out.nrrd
f.nrrd m.nrrd
```

It is obvious the immediate complexity a researcher faces when trying to process multiple "moving" images against a "fixed" one using multiple parameter set values (download moving images, iterate over them and over the parameter values, check for temporary storage space, save resulting images in temporary storage space, upload resulting images, persist execution logs, etc).

GWE Solution

Using GWE, a researcher only needs to submit a workload description (utilizing a P2EL command); which has the same general form as the associated OS shell command (above) using the generic GWE command lines clients (or an equivalent workload description using other tools such as GSlicer3 described later in this paper). The following P2EL command illustrates a real use case; which are equivalent to 125 actual analyses per moving file found in the specified location:

```
${ITER}=[10..50||10] ${HIST}=[20..100||020] ${SAMP}=[500..5000||1000]
${MOV}=f:expand(sftp://srcHost/srcDir/moving-*.nrrd)
Slicer3 --launch /usr/Slicer3/lib/Slicer3/Plugins/BSplineDeformableRegistration
--iterations ${ITER} --gridSize 5 --histogrambins ${HIST} --spatialsamples ${SAMP}
--maximumDeformation 1 --default 0
--resampledmovingfilename f:out(sftp://destHost/destDir/out-f:sys(ITER_ID).nrrd)
f:in(f.nrrd,http://otherSrcHost/otherSrcDir/fixed.nrrd?view=co) f:in(m.nrrd,${MOV})
```

GWE Solution Details

This command instructs GWE to run Slicer's "BSpline Deformable Registration" for each moving image that matches the wildcard pattern 'sftp://srcHost/srcDir/moving-*.nrrd' against the fixed image 'http://otherSrcHost/otherSrcDir/fixed.nrrd?view=co', for each possible combination of values for: iterations (ITER = 10, 20, 30, 40 and 50), histograms (HIST = 020, 040, 060, 080 and 100) and samples (SAMP = 0500, 1500, 2500, 3500 and 4500). The output is saved under the remote directory 'sftp://destHost/destDir/' with the name 'out-[ITER_ID].nrrd', where '[ITER_ID]' is the unique auto-generated identifier of the job. Transparently, GWE carries on many implicit multithreaded tasks such as:

- Determines user authentication information, and requests the user to login if SSH keys are not utilized, to the remote file system in order to be able to query it.
- Queries the remote file system to resolve the files that match the pattern used in variable ${MOV}.
- Expands the command into a set of inter-independent jobs to be processed in parallel using every possible combination of the ${ITER}, ${HIST}, ${SAMP} and ${MOV}. Each one of those jobs is a Slicer "BSpline Deformable Registration" invocation.
- Creates a virtual file system in the home directory of the user submitting the command to store downloaded files, resulting files, files to upload, log files, etc.
- When a compute node is assigned to process a job, the GWE agent scheduled on that node by the cluster's scheduling framework: downloads a copy of its requiring files to the virtual file system cache (if a copy is not already downloaded) and creates a working copy of it for the particular job; creates a placeholder file in the virtual file system for the files to upload (f:out); uploads the necessary files when finished processing; updates job status, save logs and results in DB, cleans up virtual file system; and logs for each of the processes executed (job) are saved in GWE's internal DB, so researchers can retrieve them, and output files are stored in the locations specified.

The previous task breakdown is an overview of what happens within a GWE system, however, we feel it is a good a description detailed enough to provide the reader with a feeling of the high level tasks GWE carries out on their behalf in order to get the processes executed.

3.3 GWE Generic Client Applications

The above example details the use of GWE through its generic client implementation. Two such clients exist. The GWE Terminal is a console application which keeps a live connection to a particular GWE daemon and allows the user to interactively query status information and submit requests. This application is ideal when the user is going to interact for a while with a GWE daemon. Alternatively, GWE Commands are Java command line applications meant to give the end user the capabilities to access a particular daemon, issue a particular command and exit. Usage of these applications is slightly more expensive than using the GWE Terminal since a brand new network connection has to be established every time they are invoked. However, they provide quick command line access to a GWE daemon and a basic API for integrating GWE requests programmatically through scripts.

3.4 GWE-Slicer3 Integration (GWE Slicer3 Client Application)

3.4.1 Slicer3

The National Alliance for Medical Image Computation's (NA-MIC) Slicer3[4] (http://slicer.org/) is a "free, open source software package for visualization and image analysis. Slicer's capabilities include: interactive visualization of images, manual editing, fusion and co-registering of data, automatic segmentation, analysis of diffuse tensor imaging data, and visualization of tracking information for image-guided procedures. Some of the core functionality that enables these applications include the capability to save and restore scenes using a format called MRML, a plug-in architecture to interface to external programs including ITK, a sophisticated statistical classification environment based on the EM algorithm, capabilities for rigid and non-rigid data fusion and registration, and processing of DTI MRI data."[5]

3.4.2 Objective

Researchers often encounter the need to run medical imaging algorithms over a large set of values to be permutated for different arguments (see P2EL example above), from algorithm calibration parameters to sets of images (i.e. sets of images utilized for testing an algorithm to large collections of images collected as part of a study that are ready to be processed). Executing all these processes on a single computer may take a really long time depending on the amount of processes and the number of file transfers. In addition, the gathering of results requires a lot of manual work and is highly prone to error.

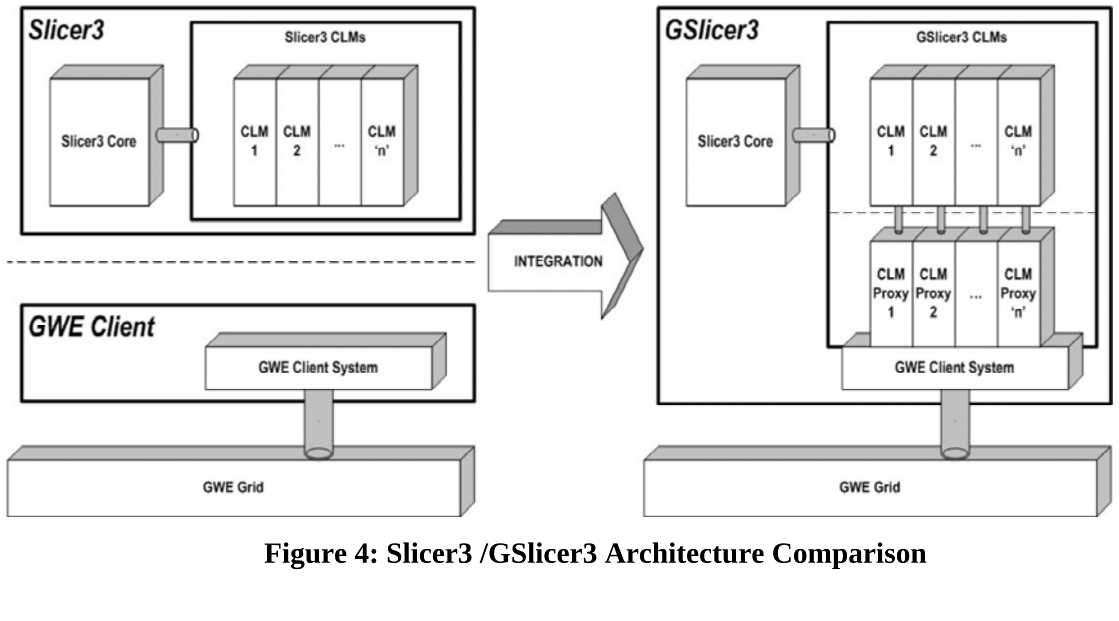


Figure 4: Slicer3 /GSlicer3 Architecture Comparison

Slicer3, provides an infrastructure to easily integrate medical imaging applications (‘modules’) as self-describing pluggable components. Taking advantage of this feature, GWE can be generically integrated with Slicer3 to allow researchers to execute their set of processes in parallel across a distributed environment while handling all the ‘side’ tasks that otherwise would have to be done manually.

3.4.3 Design

Slicer3 modules execute as regular command line applications, which must conform to a straightforward, proprietary specification. This specification requires these modules to “self describe” when invoked with the predefined reactor argument of ‘--xml’ which results in the generation of a proprietary XML descriptor stating the module’s arguments metadata (flags, types of values, label, etc). Slicer3 uses this metadata to dynamically render an appropriate UI to collect the values for each argument and to generate the module command line invocation when needed to run the module per a user’s request.

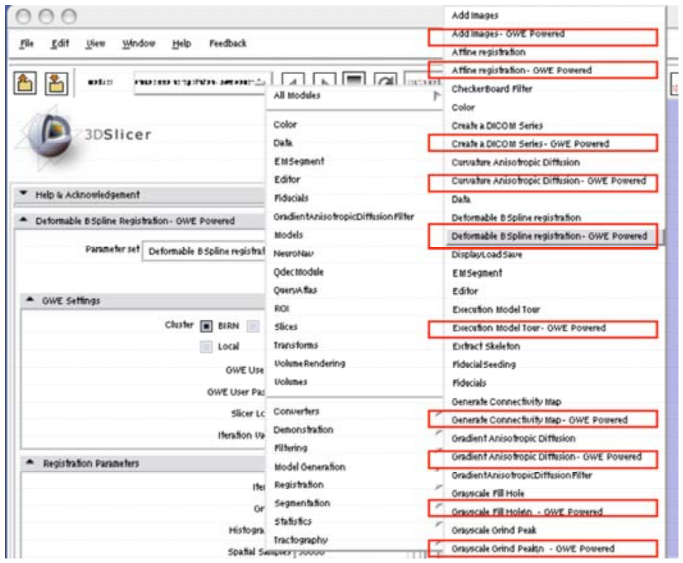
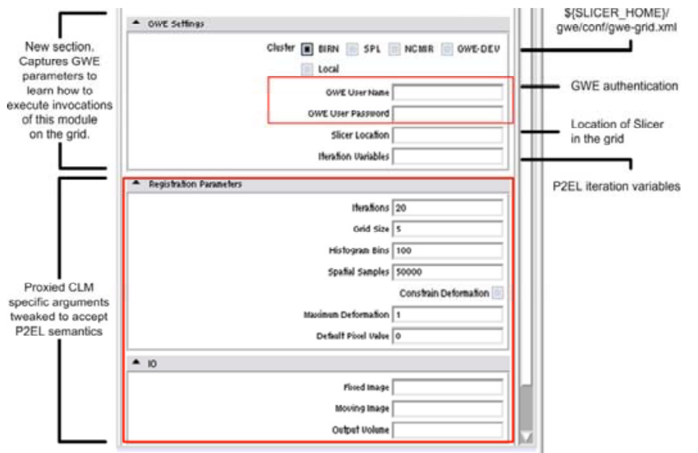

File Edit View Window Help Feedback
3DSlicer
Help & Acknowledgement
Deformable B Spline Registration - GWE Powered
Parameter set
GWE Settings
Cluster BIRN
Local
Registration Parameters
All Modules
Color
Data
EMSegment
Editor
Fiducials
GradientAnisotropicDiffusionFilter
Models
NeuroNav
QdecModule
QueryAtlas
ROI
Slices
Transforms
VolumeRendering
Volumes
Converters
Demonstration
Filtering
Model Generation
Registration
Segmentation
Statistics
Tractography
Add Images
Add Images - GWE Powered
Affine registration
Affine registration - GWE Powered
CheckerBoard Filter
Color
Create a DICOM Series
Create a DICOM Series - GWE Powered
Curvature Anisotropic Diffusion
Curvature Anisotropic Diffusion - GWE Powered
Data
Deformable B Spline registration
Deformable B Spline registration - GWE Powered
DisplayLoadSave
EMSegment
Editor
Execution Model Tour
Execution Model Tour - GWE Powered
Extract Skeleton
FiducialSeeding
Fiducials
Generate Connectivity Map
Generate Connectivity Map - GWE Powered
Gradient Anisotropic Diffusion
Gradient Anisotropic Diffusion - GWE Powered
GradientAnisotropicDiffusion Filter
Grayscale Fill Hole
Grayscale Fill Holein - GWE Powered
Grayscale Grind Peak
Grayscale Grind Peakin - GWE Powered

Figure 5: Grid enabled modules

The integration effort consists of installing the ‘GWE Client API’ inside a Slicer3 installation and generating new Slicer3 modules (Figure 5), one for each ‘regular’ Slicer3 module intended to be leveraged with grid computing capabilities. These new modules are called “GWE CLMPs” (command line module proxies) and the user can utilize them when trying to run a set of processes in a distributed grid environment, otherwise they can work as usual with the ‘regular’ versions. This effort includes a ‘bundling’ utility, which installs GWE inside a Slicer3 distribution, introspects it to discover all its pluggable modules and dynamically generates a corresponding GWE CLMP for each of them. The end result is the grid-enabled version of Slicer3, which we will refer to as GSlicer3.

When GSlicer3 is launched, the end user will notice that (in the available modules menu) for every regular module, there is another named exactly the same with the additional suffix of “ – GWE Powered” (Figure 6). These correspond to the CLMPs modules that have been auto generated by the ‘bundling’ utility. GWE CLMPs are intelligent agents, which conform to the Slicer3 module specification responding as required to the predefined reactors and have an explicit association to their corresponding ‘regular’ version module. The proxied module creates its “self description” by retrieving the “self description” of the associated ‘regular’ module and enhances them with the appropriate GWE related descriptions. Using these dynamically generated “self descriptions”, GSlicer3 is able to render a suitable UI (Figure 6) to capture the

Figure 6: Grid enabled module UI



necessary arguments to execute invocations of the associated module via GWE. Also, such UIs support P2EL semantics so value added functions (file transfers, wildcard resolution, etc) are all available. A CLMP is in essence a "GWE client application" which at runtime will carry on the following tasks:

- Retrieve, add and modify the XML tags of their proxied CLM's XML descriptor in order to add arguments, add fields to set, and capture specific GWE parameters and P2EL value types.
- Generate a GWE order with the P2EL command corresponding to the user's input appropriately translated to the selected cluster resource (e.g. Slicer3 location, user home directory, etc).
- Install a customized Slicer result parser to the GWE order.
- Submit the GWE order created to the selected cluster resource, over the secured RPC GWE network.
- Monitor, in real time, the execution progress of the localized proxied CLM invocations (from the GWE order) on the selected cluster. This real time monitoring is also performed over the secured RPC GWE network.
- Keep track of the CLMP progress as a ratio of the number of proxied CLMs invocations already executed divided by the total amount of proxied CLMs invocations associated with the GWE order submitted.
- Notify Slicer3 of this progress using Slicer3 XML based progress API (<filter-XXX > tags sent to the standard output of the CLMP).

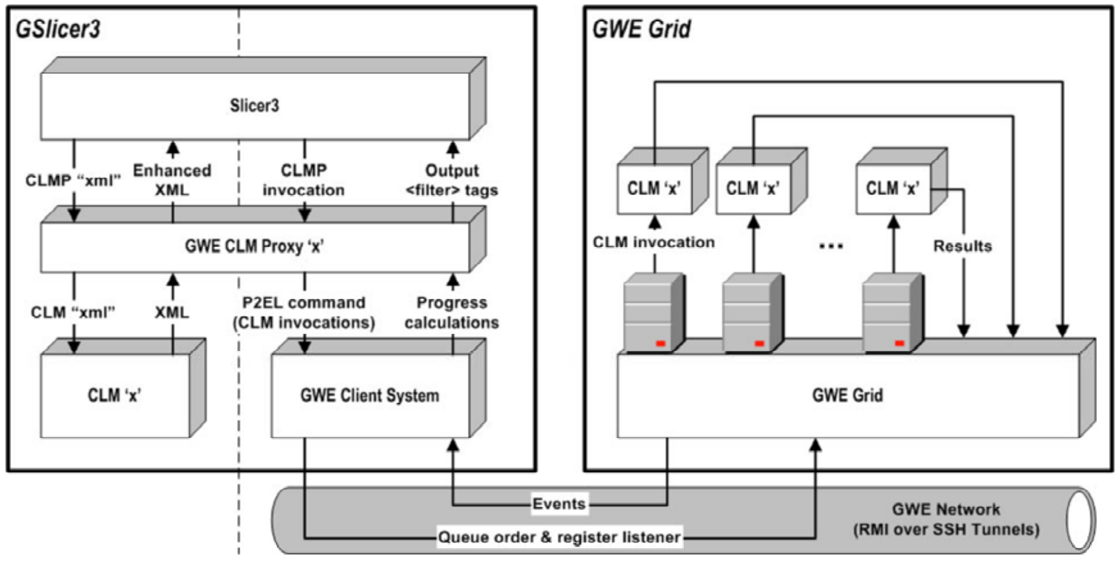


Figure 7: GSlicer3 Execution Flow

This integration effort provides a generalized, easy to use grid computing enabled interface to all Slicer3 CLMs that are "Standard Execution Model" compliant.

3.4.4 Usage

As described in the previous section, every GWE CLMP has an associated UI where the user can input values for its parameters (Figure 6). This UI supports the input of parameters expected by its associated 'regular' version module and/or P2EL semantics (see P2EL section). Using these values, the GWE CLMP generates a P2EL command and submits it to the selected cluster (Figure 7). The selection of the cluster is done by selecting one of the possible clusters, defined in the gwe-grid.xml file (Section 3.1), shown as radio buttons on top of the UI. For example, using the "BSpline Deformable Registration" CLMP a user can submit the order described in section 4.2 via the GSlicer3 user interface.

4 Future Work

This system is currently in its 4th alpha release. The current project plan is to release a new version every 6 weeks and release the first feature complete beta version in the beginning of 2009. In order to reach the beta release, the product is going through extended testing with biomedical researchers to elicit usability and functional requirements. As a result of this testing the following features are being finalized (a-d), implemented (e-h) and tested:

a. Application registry framework - provides capabilities to auto-deploy applications to running clusters on an as needed basis. Currently the Slicer3 integration described above assumes that Slicer3 is installed on the running clusters.
b. Multi-cluster module - provides true grid abstraction through the ability of GWE to distribute jobs to a specified set of clusters in a “daisy chain” configuration.
c. Array of iteration variables feature - provides the means to specify a set of values-sets to apply all at once as a single variable. For example: VAL_SET=[(0,10,20),(15,27,-3)] would create 1 permutation with the first set of 3 values and a second permutation with the second set.
d. Parametrical order behavioral logic - provides the means for end users to customize execution aspects of the jobs associated with an order, such as, job timeouts, launch mode, file system clean up policy, maximum concurrent jobs running, etc.
e. Alert notification module - provides the means for end users to specify job runtime conditions under which the system shall send notifications to customizable recipients (typically an email to a specified email address).
f. Job result parser framework - provides an infrastructure for the end user to create his/her own result parser to inject into a specific order so it can extract meaningful structured data out of the job results.
g. Additional grid related drivers for file system drivers (i.e. SRB, XCEDE based XML file catalogs, and GridFTP) and resource managers (e.g. Torque).
h. Portal client integration as JSR-168 portlets that can be deployed to any standards based portal container.

References

1. J. Epema, M. Livny, R. Dantzig, X. Evers, and J. Pruyne. A worldwide flock of condors. Journal on Future Generations of Compute Systems , 12, 1996.
2. I. Foster and C. Kesselman. Globus: A Toolkit-Based Grid Architecture, pages 259–78. Morgan Kaufmann, 1999.
3. W. Gentzsch. Sun grid engine: Towards creating a compute power grid. Cluster Computing and the Grid, 00:35, 2001.
4. S. Pieper, B. Lorensen, W. Schroeder, and R. Kikinis. The NA-MIC Kit: ITK, VTK, Pipelines, Grids and 3D Slicer as an Open Platform for the Medical Image Computing Community, Proc IEEE Intl Symp on Biomedical Imaging 2006; 1:698-701.
5. http://slicer.org/pages/Introduction, May 2008

Simplified Grid Implementation of Medical Image Processing Algorithms using a Workflow Managment System

Dagmar Krefting, Michal Vossberg and Thomas Tolxdorff

Institute of Medical Informatics, Charité - Universitätsmedizin Berlin, Germany, dagmar.krefting@charite.de

Abstract

Complex analysis of large amounts of medical image data quickly exceeds storage capacity and computing power of single workstations or small local networks. When limited hardware resources impede full utilization of medical image analysis, a possible solution is the usage of computing grids, the collaboration of distributed resources across institutional borders. Many existing image processing problems would benefit from parallel processing, e.g. on single image or volume slice level. Such coarse-grained parallelization can easily be achieved by implementation into a grid infrastructure. Furthermore, grids allow distributed users to share their code, promoting collaborative projects. In this paper, we describe the grid implementation of existing code using grid workflows. The workflow management system is able to execute all tasks related to grid communication, such as authorization, scheduling and monitoring. It remains to the developer to make the code accessible for the workflow manager, and to define, where the code is found on the grid and what to do with it. We describe the procedure how to bring the code to the grid and show exemplarily the implementation of segmentation and registration algorithms for transrectal ultrasound guided prostate biopsies.

1 Introduction

Many scenarios in medical image processing demand high computing power and storage capacity. Increasing usage of high resolution images and multidimensional data, like volume sequences or multi-modality data, amplify hardware requirements. When results are required within a certain time, compromises between accuracy and computing time are unavoidable on limited resources. Furthermore, new algorithms developed by research groups are often hardly available or adaptable for related research problems. A promising solution to overcome these barriers are modern grid networks. A grid infrastructure is defined as a collaboration of possibly inhomogeneous, distributed hardware resources across administrative borders [4]. The network is virtualized as a single resource providing scalable hardware resources and a variety of applications. The management of the network is realized by the middleware, an abstraction software layer between the grid network and the applications.

Many medical image analysis tools process consecutively image series, volume slices or tiles, or loop over a range of input parameters, e.g. multiscale analysis. The computing time of such processing steps scales with 1/n (n = number of frames, slices, or tiles), when coarse-grained parallelized and implemented to a distribute computing system with more than n CPUs. Due to transfer times, when reuniting intermediate results for integrated analysis, the resulting compute time savings are reduced, but in most cases notable. Especially in the development state of image processing algorithms, where large parameter scans are realized, distributed

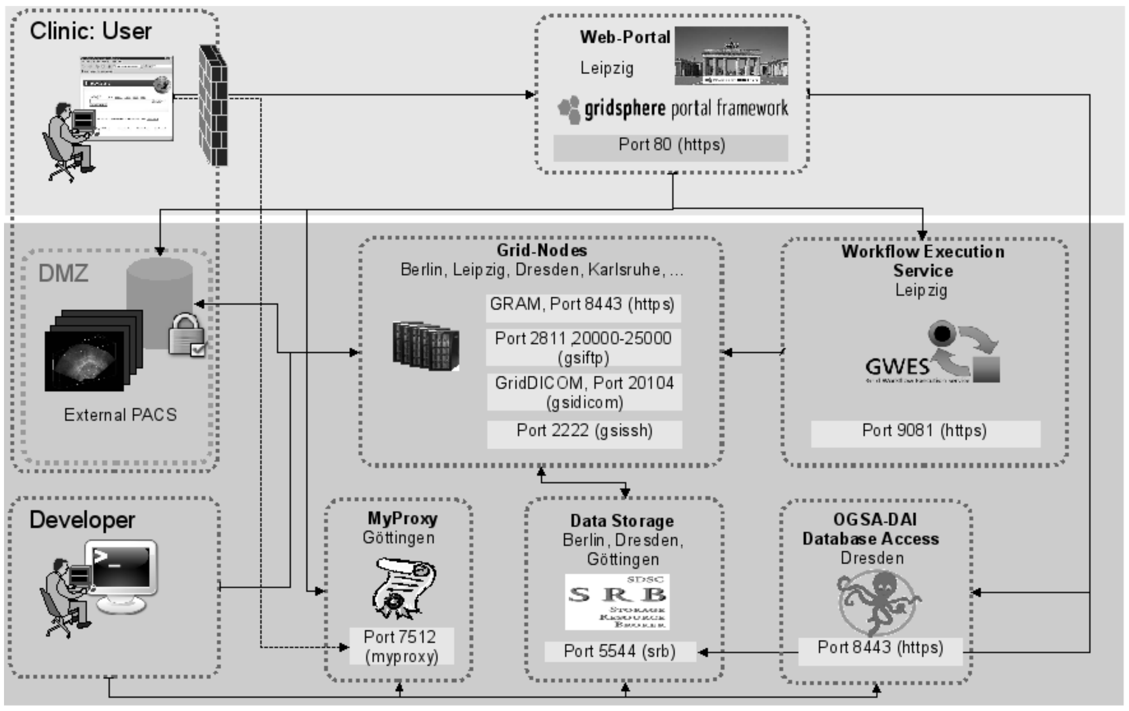


Figure 1: Implemented MediGRID system architecture.

computing may save days of waiting for results. But not only time saving aspects may be relevant when considering to use grids for medical image applications. Access to distributed image databases or distributed software with particular hardware requirements can easily be achieved. If a grid portal - a webbased user interface - is set up, users may start, stop and check grid jobs from every place with internet access. However, if the expected effort for the grid implementation is estimated to outbalance the (near-term) benefit, interested researchers are discouraged to use grid resources. The method proposed in this paper allows the grid implementation without deep knowledge of the underlying grid infrastructure. Quick results can be achieved by basic implementation using XML-based workflow descriptions. The implementation can later be enhanced regarding user friendliness by integration into a grid portal.

2 Methods

2.1 Grid architecture and middleware solutions

The implementation is realized within the German MediGRID infrastructure, which is part of D-Grid[15, 8]. D-Grid offers generic middlware components for a variety of user communities, from high energy physics to linguistics. MediGRID uses basic services from D-Grid, e.g. user management, but has extended the hard- and software to satisfy community specific requirements. It is a *Globus* (GTK4) based grid infrastructure, which is the de facto standard for modern grid architectures[7]. It provides built-in security mechanisms, data transfer (gridftp) and generic job submission (GRAM). A grid portal, based on *GridSphere*, allows for easy access from different sites even behind strict firewalls[19]. The main extensions above the core infrastructure are the Grid Workflow Execution Service (GWES)[10], the Storage Resource Broker (SRB) for distributed data storage[21], and gridDICOM for medical image transfer[23]. The implemented system architecture of MediGRID is given in Fig. 1.

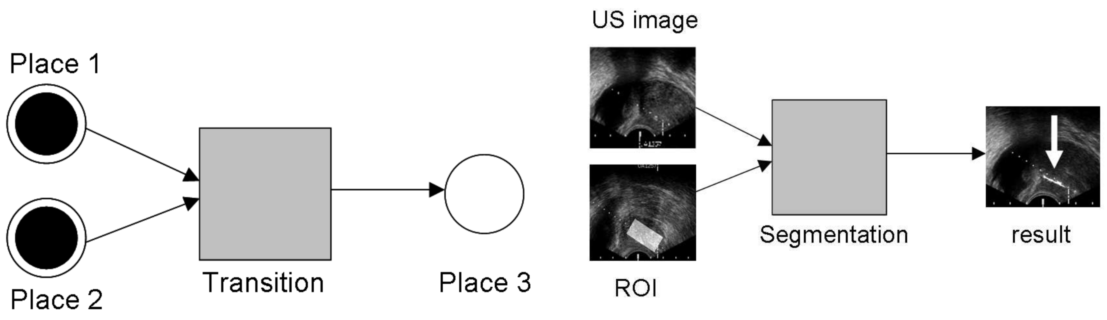


Figure 2: A simple petrinet (left), modelling the execution with two inputs and one output, e.g. biopsy needle segmentation (right).

2.2 GWES

GWES is a workflow manager established within the K-WF-grid[5]. In contrast to other scientific workflow systems, such as Condor's DAGMan tool[6] or LONI[22], it is specifically targeted for WSRF-standardized grid usage and allows to create complex workflows without detailed knowledge of the grid infrastructure. The core of the GWES is the Grid Workflow Description Language (GWorkflowDL), which is an XML based standard for describing workflows as a Petri net. A Petri net is a mathematical formalism to describe discrete distributed systems and allows to model the workflow with a few basic graph elements. As such, Petri nets are often easier to use and more intuitive than other workflow languages, such as the widely known BPEL (Business Process Execution Language)[2], which has the disadvantage of posessing complex and rather informal semantics and an extensive syntax. Other graph-based languages mostly base on directed acyclic graphs (DAGs) and offer only a limited expressiveness so that it is often hard to describe complex workflows (e.g., loops cannot be expressed directly). For the actual workflow descriptions, GWES uses an extension, high level Petri nets (HLPN). They can be directly used to model transfer, execution and storage of any kind of input and output data as well as control data (e.g. the exit status of a workflow step). Data is modeled as *tokens*, located at a *place*, and program execution is represented as a *transition*. In contrast to control-centric languages, like BPEL, GWES is data driven and the Petri net representation corresponds to the pure data flow. Additionally, the flow can be controlled through user contraints to the processor, which is a benefit over other data-flow languages like Scufl/XScufl[13].

The basic workflow description provides the definition of transitions and their input- and output places. Figure 2 gives a simple example of a Petri net. It consists of one transition, represented by a square, two input places (Place 1 and Place 2) and an output place (Place 3). Places are represented by open circles. Directed arcs connect places and transition and define the flow relation. Tokens are represented as black dots. This Petri net might - for example - model the segmentation of a biopsy needle in an ultrasound image. Then Place 1 models the US image, Place 2 models a predefined region of interest, the transition models the execution of the segmentation algorithm and Place 3 models the result. At the current initial state of the processing, tokens are located at the input places. Within runtime of the workflow, tokens on input places are consumed and sent to output places. Highlevel Petri nets can do anything that can be defined in terms of an algorithm [12]. We found them perfect to map the image pipelines of our applications. GWES descriptions can be realized on several abstraction levels, which are then concretized during runtime. It provides basic resource brokering and scheduling. The information about available hard- and software is provided in the XML-based D-Grid Resource Description Language (D-GRDL), and is stored in the resource database[1]. GWES also offers fault-tolerance strategies for reliable process execution. If an execution step fails, the error is reported and the transition is rescheduled to another resource up to an adjustable number of retrials.

The generic GWES web interface (GWUI) allows to upload of workflow descriptions and monitoring of running workflows. An automated visualisation of the workflow layout and the live status are also offered. This is particularily helpful in the development and testing phase, as workflow layout and implementation errors immediatly become visible.

3 From command line program to a grid application

In this section we describe the different steps that have to be processed by the developer to bring an existing image processing program onto the grid. We will focus here on automated image processing steps to demonstrate the basic implementation method. Moderate user interaction like workflow suspension and restarting is provided by the GWES webinterface. Further usability can be realized completely webbased by integration of the workflow into the grid portal. The grid implementation encompasses the following steps:

1. Deployment of the software to the gridnodes
2. Generation of a wrapper script
3. Registration of the software
4. Creation of a workflow description
5. Optional: Integration of the workflow into the user portal

3.1 Deployment of the software

The easiest way to make software available on the grid is to deploy it as appropriatly compiled code. All software related to the specific application is stored at a separate directory on each frontend of the connected clusters in the used infrastructure. The frontends are accessible via *gsissh*, a grid enabled version of ssh. Developers can log on to these machines and manage their application directories. As of today, there is no effective automatic deployment of the software. We use a version manager (subversion) to ease the update process of the grid nodes[3].

3.2 Writing GWES wrapper scripts for the software

During workflow execution, the GWES workflow manager calls the programs designated by the transitions of the workflow's Petri net. In the call, the name of the transition denotes the executable and the connected input tokens specify the input parameters. As the tokens in the net adhere in no particular order, GWES passes the parameters as key/value pairs and uses the edge-expression of the token's arc as the key (i.e. *program-name -edge_expr1 token1 -edge_expr2 token2 ...*). Since most existing programs do not conform to this plugin convention, it proved good pratice for the developer to provide a small wrapper script which is called instead of the actual program. In the wrapper script the parameter values are extracted from the key/value pairs and mapped to the application specific program call. This also adds an additional layer of abstraction, as the script can be changed more easily than the workflow or may contain auxiliary commands.

3.3 Registration to the resource database

To publish the new software to the grid, a resource description file is generated and stored in the database. The resource description contains the internal GRDL name of the software and the path of the wrapperscript

```
<resource  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation=”http://www.gridworkflow.org/kwfgrid/src/xsd/resource-d-grdl.xsd”
  uri=”software:medigrid-wrapperscript”>
<ofClass uri=”urn:dgrdl:software:medigrid-wrapperscript”/>
 <name>wrapperscript</name>
 <description>Medical image application: Example of a wrapper script</description>
 <simpleProperty  ident=”executable”  type=”string” unit=””>
  /wpath/wrapperscript.sh
 </simpleProperty>
</resource>
```

Figure 3: Example of a GRDL resource description.

on the gridnode. An example for the script *wrapperscript.sh*, located at */wpath* is given in fig. 3. An XML-element, containing the URI of the software, is added to the hardware resource description files to let the system know where the software is available.

3.4 Creation of a workflow description

The workflow description is also an XML document, using GWorkflowDL. This part of the implementation might need some familiarization for user who are inexperienced with XML. The workflow definition consists of the following parts

1. Header: provides properties and execution information for the GWES.
2. Place elements: defines places and initial assignment with tokens.
3. Transition elements: defines the software to be executed and its input and output places.

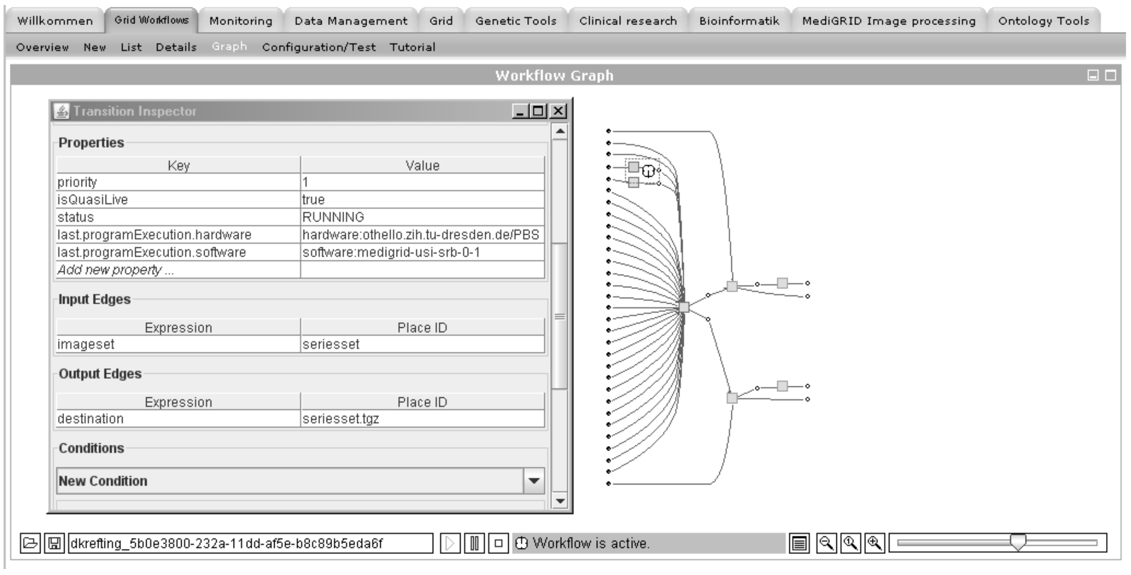
An example can be found in MediGRID project's deployment guide[11].

3.5 Visualization and execution of a workflow

The workflow description can now be uploaded to GWES using the web interface, where the underlying Petri net is visualized. If the resulting layout is as expected, the workflow can be started. The actual state of the places (occupying tokens) and transitions (available and chosen hardware resource, state) as well as possible error messages or warnings are monitored. The upload of the workflow description is the default for development and testing. The disadvantage is, that the initial tokens have to be defined directly in the workflow description. This is fine for quick implementation, if the programs are used by the developer. A more userfriendly method is to integrate the generation of the workflow into a graphical user interface.

3.6 Using workflow templates

Graphical user interfaces are implemented mainly as grid portlets within the used grid infrastructure[14]. When integrating the workflow into such a GUI, users can upload data or select them from available data storage (PACS and SRB). Additional parameters can be set or modified. The desired image processing workflow can be started at the push of a button. A workflow template stored within the portal is then automatically complemented and transferred to the GWES. This is the most convenient solution for the end user, but requires of course more development time for writing the grid portlets[20].



Properties

Key	Value
priority	1
isQuasiLive	true
status	RUNNING
last.programExecution.hardware	hardware:othello.zih.tu-dresden.de/PBS
last.programExecution.software	software:medigrid-usi-srb-0-1
Add new property ...	

Input Edges

Expression	Place ID
imageset	seriesset

Output Edges

Expression	Place ID
destination	seriesset.tgz

Conditions

New Condition

dkrefting_5b0e3800-232a-11dd-af5e-b8c89b5eda6f

Workflow is active.

Figure 4: Single registration workflow with status control.

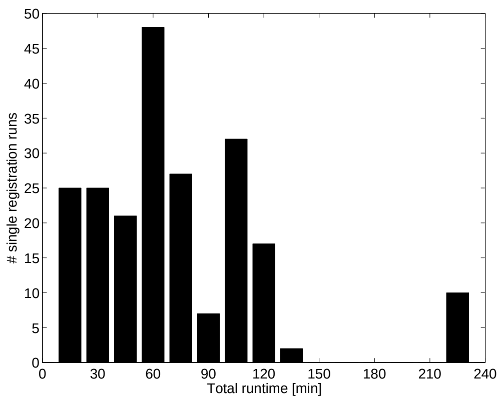


Figure 5: Runtime distribution of 214 single registration workflows.

4 Implementation of image processing algorithms

To date, the described method was used to successfully implement completely different applications to the grid, as parallelized preprocessing of functional MRI data, hemodynamic simulations, gene prediction or biosignal analysis of polysomnographies. Here, we present the implementation of algorithms to be used for analysis of transrectal ultrasound images taken during prostate biopsies. Prostate cancer is the most common cancer in men. Current goldstandard for prostate cancer diagnosis is ultrasound guided prostate biopsy[9]. Monitoring the prostate by transrectal ultrasound (TRUS), tissue probes are taken from different parts of the prostate. The present application determines and visualizes the position of the tissue probes within the prostate volume. The localization is done by automated segmentation of the biopsy needle in the guiding 2D ultrasound images and subsequent registration of the 2D images into a previously taken 3D ultrasound volume[16]. The algorithms are developed in *Matlab* and *c/c++* using *ITK,VTK* and *MITK*[17, 18]. While segmentation and registration are mainly independent, the respective workflows are modeled separately. 2D-3D image registration of the transrectal ultrasound images turned out to be a difficult task, as many local minima exist in the used costfunctions (normalized correlation and mutual information). Furthermore,

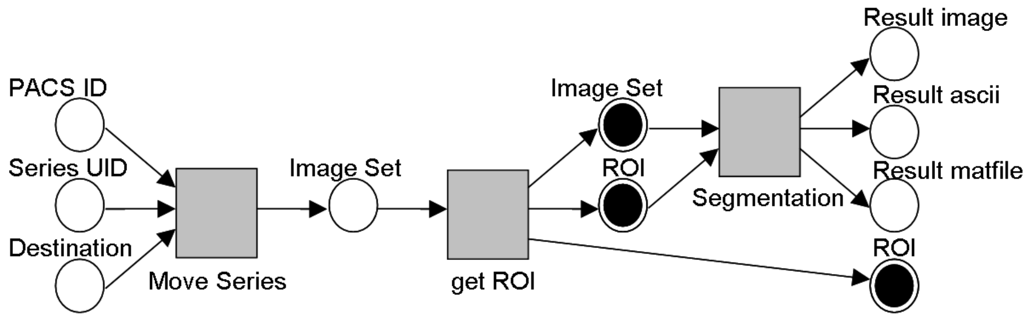


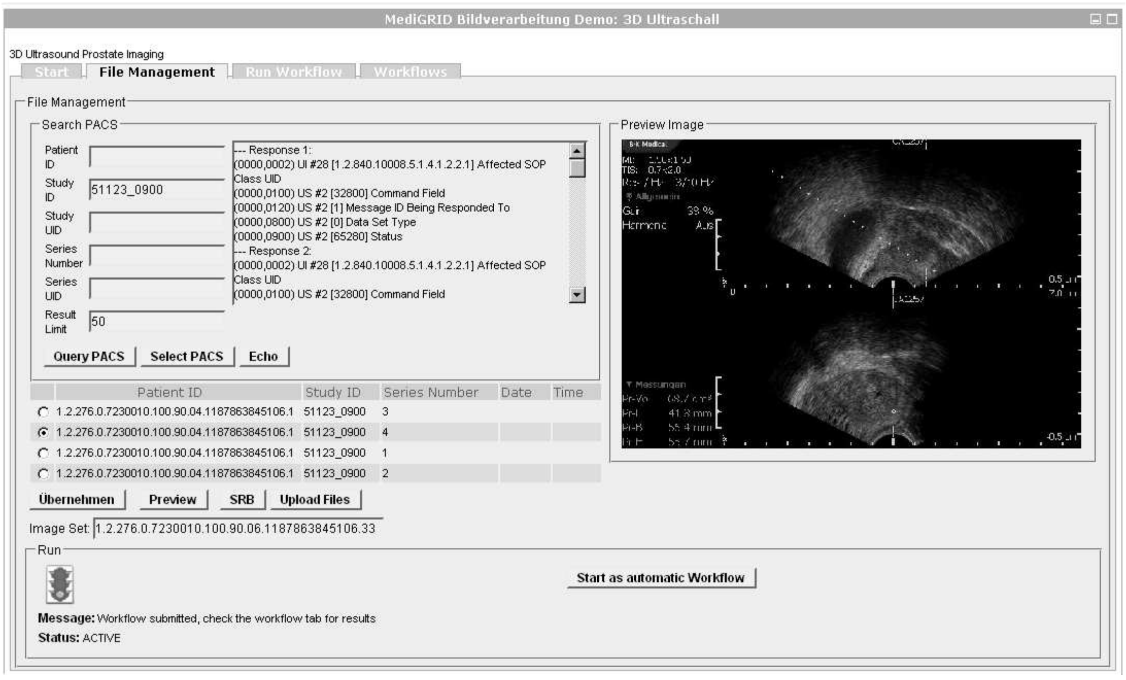
Figure 6: Petri net of the segmentation workflow. Image transfer and ROI calculation are already finished, the segmentation will be executed next.

the optimization algorithms implemented in *ITK* show poor performance in optimization of rotation angles. Therefore a parameter scan of up to 600 single registration runs with different inital rotation angle values is performed. A single registration run for maximum allowed 1000 iteration steps needs in average 20 minutes. As an example, our workstation (AMD 64 X2 Dual Core Processor 4200+, 2Gb RAM) requires a processing time of about 4 days to complete the scan. The single registration run on the grid can be devided in the following processing steps:

1. Volume and 2D image transfer from SRB.
2. Registration execution.
3. Storage of the results in SRB.
4. Download of the results.

Fig. 4 shows the workflow layout and the status information for the registration workflow provided by the GWES web interface. The many input places of the central transition, which denotes the registration execution, are the variety of additional parameters which are passed to the algorithm, e.g. the costfunction and optimizer used for registration. With grid implementation, we were able reduce the total runtime down to 12 hrs, which is approximately 8 times faster than using the workstation. A detailed look to a scan consisting of 214 initial parameter sets gives more insight into the performance. Fig. 5 shows the distribution of total runtimes of the registration runs. Total runtime includes the waiting time before scheduled, the queue time on the selected gridnode and the CPU time. The total runtime reaches from 20 minutes up to 220 minutes (3.7 hrs), which determines the total runtime of the complete scan. The difference between the ”fastest” and the ”slowest” single registration run is mainly waiting time due to high load on the gridnodes, which can be accessed by all users of the D-Grid and is extensively used for jobs running approximately 12 hrs. The average load on the sites used, encompassing 580 cpus, was between 80% and 100% during the runtime of the workflows. The completion time was more than 9 times faster than the approximate runtime of 35 hrs on our workstation, but still a factor of 10 of the “optimum” of 20 minutes, which would be the pure cpu-time with vanishing queue times. Current expansion of the grid resources and implementation of advanced local scheduling algorithms are let us expect further reduction of the overall runtime. At the current state of the infrastructure, failure rates of up to 5% are experienced. The relatively high number of failures is mainly due to the fact, that the infrastructure itself is still under development and middleware components are restarted frequently.

The needle segmentation consists of the following steps:



Patient ID	
Study ID	51123_0900
Study UID	
Series Number	
Series UID	
Result Limit	50

```
--- Response 1:
(0000,0002) UI #28 [1.2.840.10008.5.1.4.1.2.2.1] Affected SOP
Class UID
(0000,0100) US #2 [32800] Command Field
(0000,0120) US #2 [1] Message ID Being Responded To
(0000,0800) US #2 [0] Data Set Type
(0000,0900) US #2 [65280] Status
--- Response 2:
(0000,0002) UI #28 [1.2.840.10008.5.1.4.1.2.2.1] Affected SOP
Class UID
(0000,0100) US #2 [32800] Command Field
```

	Patient ID	Study ID	Series Number	Date	Time
○	1.2.276.0.7230010.100.90.04.1187863845106.1	51123_0900	3		
●	1.2.276.0.7230010.100.90.04.1187863845106.1	51123_0900	4		
○	1.2.276.0.7230010.100.90.04.1187863845106.1	51123_0900	1		
○	1.2.276.0.7230010.100.90.04.1187863845106.1	51123_0900	2		

Image Set: 1.2.276.0.7230010.100.90.06.1187863845106.33

Message: Workflow submitted, check the workflow tab for results

Status: ACTIVE

Figure 7: Screenshot of the segmentation portlet. Input image sequence can be selected from the PACS, the SRB, or can be uploaded. After image selection, the workflow can be initiated.

1. Image retrieval from a PACS.
2. Calculation of the Region of Interest (ROI) on the first image of the series.
3. Segmentation of the image series.

The workflow layout is given in Fig. 6. The code was written in *Matlab* and was compiled using Matlab's *Compiler Toolbox*. The time saving of grid usage is here on image sequence level. Each biopsy procedure encompasses 10 individual tissue probes at different prostate locations which are then sent to different grid nodes, depending to the actual load. Since the segmentation itself only needs a few minutes and transfer load would be quite high (each single image needs the ROI and the statistical model), parallelization on image level was abandoned in the segmentation. An application specific portlet was developed. The image series can now be selected interactively and the results are visualized on the grid portal. A screenshot of the portlet is shown in Fig.7.

5 Discussion

The use of the workflow manager makes it easy to integrate existing code without deeper knowledge of grid computing and the underlying system architecture. Automated medical image processing problems with high coarse grained paralellization potential (parameter scans, image sequences) can profit from grid use with low implementation offset. Basic implementation of new algorithms can be performed from developers even with little experience in less than a day. If the applications are used repeatedly or shall be available to end users, integration of workflow templates into application specific grid portlets is strongly recommended. A portlet with basic functionality may be developed within 2 days, but advanced user interaction may need development time up to a few weeks. The additional effort is rewarded by ease of use and full application control from all sites with internet access, even from institutions with strict firewall settings, e.g. clinical environments. The presented grid infrastructure is mainly used for research purposes. Besides implementation of further applications, current developments focus on security, reliability and usability to allow grid use for clinical applications.

6 Acknowledgements

The work is funded by the German Federal Ministry of Education and Research (BMBF), grant number 01AK803F.

References

[1] Martin Alt, Andreas Hoheisel, Hans-WernerPohl, and Sergei Gorlatch. A grid workflow language using high-level petri nets. In *Proceedings of the 6th international conference on Parallel processing and applied mathematics. PPAM 2005. Poznan*, pages 715–722, 2005.

[2] T. Andrews, F Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services (BPEL4WS). Specification Version 1.1. Technical report, Microsoft, BEA, and IBM, 2003.

[3] B. W. Fitzpatrick B. Collins-Sussman and C. M. Pilato, editors. *Version Control with Subversion*. O'Reilley, 2004. http://svnbook.red-bean.com/.

[4] V. Breton, A.E. Solomonides, and R.H.McClatchey. A perspective on the Healthgrid initiative. In *4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2004), Chicago, Illinois, USA*, pages 434–439. IEEE Computer Society, 2004.

[5] M. Bubak, T. Fahringer, L. Hluchy, A. Hoheisel, J. Kitowski, S. Unger, G. Viano, K. Votis, and K-WfGrid Consortium. K-Wf Grid - Knowledge based Workflow system for Grid Applications. In *Proceedings of the Cracow Grid Workshop 2004*, page 39. Academic Computer Centre CYFRONET AGH, 2005.

[6] Condor Team. DAGMan: A Directed Acyclic Graph Manager, July 2005. http://www.cs.wisc.edu/condor/dagman.

[7] Ian Foster. Globus toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing*, pages 2–13. Springer-Verlag LNCS 3779, 2005.

[8] W. Gentzsch. D-grid, an e-science framework for german scientists. In *Proceedings of The Fifth International Symposium on Parallel and Distributed Computing (ISPDC 2006)*, pages 12–13. IEEE Computer Society, 2006.

[9] A. Heidenreich, G. Aus, C. C. Abbou, M. Bolla, S. Joniau, V. Matveev, H-F. Schmid, and F. Zattoni. Guidelines on prostate cancer, Update March 2007. http://www.uroweb.org/fileadmin/user_upload/Guidelines/Prostate%20Cancer.pdf.

[10] A. Hoheisel. Grid workflow execution service - dynamic and interactive execution and visualization of distributed workflows. In *Proceedings of the Cracow Grid Workshop 2006*, pages 13–24. Academic Computer Centre CYFRONET AGH, 2007.

[11] A. Hoheisel, D. Sommerfeld, and K. Peter. *Guidelines for the Deployment of Applications in MediGRID*, December 2007. http://www.medigrid.de/downloads/MediGRID_application_deployment_1-2.pdf.

[12] Andreas Hoheisel and Martin Alt. Petri nets. In Ian J. Taylor, Dennis Gannon, Ewa Deelman, and Matthew S. Shields, editors, *Workflows for e-Science - Scientific Workflows for Grids*. Springer, 2006.

[13] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole Goble, Mathew R. Pocock1, Peter Li, and Tom Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Res*, 34:729–732, 2006.

[14] JavaCommunityProcess. JSR-000168 Portlet Specification, 2005. http://jcp.org/aboutJava/communityprocess/review/jsr168/index.html.

[15] D. Krefting, J. Bart, K. Beronov, O. Dzhimova, J. Falkner, M. Hartung, A. Hoheisel, T. A. Knoch, T. Lingner, Y. Mohammed, K. Peter, E. Rahm, U. Sax, D. Sommerfeld, T. Steinke, T. Tolxdorff, M. Vossberg, F. Viezens, and A. Weisbecker. Medigrid: Towards a user friendly secured grid infrastructure. *Future Generation of Computer Systems, doi:10.1016/j.future.2008.05.005*, 2008.

[16] D. Krefting, B. Haupt, and T. Tolxdorff. Segmentation of prostate biopsy needles in transrectal ultrasound images. In *Proc. SPIE 6512*, pages 6512–105, 2007.

[17] Mathworks, Inc. MATLAB - The Language of Technical Computing, 2006. http://www.mathworks.com.

[18] NLM. Insight Segmentation and Registration Toolkit (ITK), 2006. http://www.itk.org.

[19] Jason Novotny, Michael Russell, and Oliver Wehrens. Gridsphere: An advanced portal framework. In *Proceedings of the 30th Euromicro Conference*, pages 412–419, 2004. http://www.gridsphere.org.

[20] GridSphere Project. *Grid portlets development guide*. http://docs.gridsphere.org.

[21] A. Rajasekar, M. Wan, R. Moore, W. Schroeder, G. Kremenek, A. Jagatheesan, C. Cowart, B. Zhu, Sheau-Yen Chen, and R. Olschanowsky. Storage Resource Broker - managing distributed data in a grid. *Computer Society of India Journal*, 33(4):42–54, Oct 2003.

[22] David E. Rex, Jeffrey Q. Ma, and Arthur W. Toga. The LONI pipeline processing environment. *NeuroImage*, 19:1033–1048, July 2003.

[23] Michal Vossberg, Thomas Tolxdorff, and Dagmar Krefting. DICOM Image Communication in Globus-Based Medical Grids. *IEEE Trans. Inf. Technol. Biomed.*, 12:145–153, 2008.

From gridified scripts to workflows: the FSL Feat case

Tristan Glatard[1,2,3] and Sílvia D. Olabarriaga[1,2,3]
[1]Radiology, [2]Clinical Epidemiology, Biostatistics and Bioinformatics
Academic Medical Centre, University of Amsterdam, NL
[3]Institute of Informatics, University of Amsterdam, NL

Abstract

In this paper we investigate the benefits of moving from monolithic script implementations to workflows for performing functional magnetic resonance imaging (fMRI) experiments on the grid. In particular, we focus on the case of the FSL fMRI Expert Analysis Tool (*Feat*). Concrete pros and cons of both approaches are provided to fMRI analysts, and lessons learned from the study of a quite complex grid workflow use-case are reported to grid scientists. Both usability and performance are studied. The problem of output presentation is described in details. Simulations results show that the performance improvement yielded by the workflow implementation in a job farming experiment is limited, whereas it is significant for parameter sweeps. Even if hard-coded ad-hoc solutions can yield benefits for some specific use-cases, using workflows for such a widely scoped application is still challenging in terms of description, usability and performance.

1 Introduction

Experiments performed in functional magnetic resonance imaging (fMRI) are nowadays reaching a scale that motivates the adoption of grid technology to address performance and experiment management issues [9, 1]. Meanwhile, workflow has emerged as a paradigm for gridifying applications in medical image analysis [10] as in several other scientific fields [3, 2]. In this paper we discuss the implementation of fMRI analysis using a workflow approach.

The FMRIB software library (FSL) [12], Statistical Parameter Mapping (SPM) [4], Brain Voyager [7] and Analyis of Functional NeuroImages[1] (AFNI) are among the representative packages used for the analysis of fMRI data. In this work, we focus on the FSL fMRI Expert Analysis Tool *Feat*, a tcl script gluing together calls to several FSL command-line utilities to implement fMRI analysis. This script is highly parametrised, at the same time hiding details of the complex image analysis pipeline underneath and enabling the user to choose subsets of the analysis to be performed in a particular invocation. Nevertheless, it is fair to say that the Feat application as a whole is seen by the users as a “monolithic program” that is invoked with a large list of parameters grouped into the “design file”. One can think of at least two advantages of moving from such a monolithic to a workflow approach: usability and performance. *Firstly*, workflows allow more flexibility to the application. For instance, replacing part of the application (e.g., a registration method) for

[1]`http://afni.nimh.nih.gov/afni`

experimental purposes is easier with a workflow. Grid workflows also improve application management such as version maintenance (through the use of remotely maintained components) or error detection and handling. Moreover, workflow descriptions are also more suitable for execution on grids, as parallelism is naturally expressed. *Secondly*, workflows are expected to yield gains in terms of execution time due to exploitation of intrinsic parallelism in a trivial way (and without user intervention). Performance improvement may come from computation savings, in particular in parameter sweep experiments. In monolithic implementations, the whole analysis is computed for each parameter combination, even if some steps do not depend on the varied parameters (such as performed in [9]), whereas such a redundancy is naturally avoided with workflows.

Based on the experience of porting a representative application as a workflow, this paper discusses concrete pros and cons about moving from a script-based to a workflow implementation of FSL *Feat*, while reporting some lessons learned from the study of a quite complex grid workflow use-case. A workflow implementation of the *Feat* application is first described in Section 2. In the rest of the paper we then address two aspects that turned out as important in the context of this work, namely the organization of outputs generated by the workflow (Section 3) and performance analysis (Section 4). Section 5 closes the paper with preliminary conclusions of this exercise.

2 Workflow description

We focus on *Feat* "first-level" analysis, which aims at calculating brain activation maps of the fMRI scan of a subject in response to stimuli imposed during the acquisition. Using the General Linear Model (GLM), *Feat* determines the voxels of the fMRI scan whose intensity variation along time is correlated to the stimulus signals. Stimulus signals are also called *model Explanatory Variables* (EVs) and linear combinations of them are denoted *contrasts*, being used for example to study the independence of the EVs. Normalization of the fMRI scan to a standard or template brain is also performed to allow group studies in further steps ("high-level" analyses in *Feat*). We considered FSL version 3.3 for this study. Recently, the FSL developers in Oxford released version 4.0 of the software package in which Feat leverages the Sun Grid Engine to job farm across subjects.

2.1 Implementation

Figure 1 presents an impression of our implementation of the *Feat* application workflow. A valid description of the workflow in the Scufl language [8] is available from the myExperiment community web site[2]. This workflow runs on the grid infrastructure of the Dutch VL-e project[3] and its output has been checked to be identical to the one produced by the script version of *Feat*. Details about Scufl are available on the Taverna website[4]. The workflow consists of 30 grid processors corresponding to executables from FSL that were wrapped for the grid using the Generic Application Service Wrapper (GASW) [5]. Processors implementing basic functions like arithmetic operations or string concatenation were wrapped as local Java Beanshells[5]. The 41 inputs correspond to *Feat* parameters, as specified in the design file used by the script version. The 65 outputs are resulting files stored by *Feat* in a directory tree.

Three sub-workflows are identified in figure 1: normalization, pre-processing, model computation. The

[2] `http://www.myexperiment.org/workflows/176`

[3] `http://www.vl-e.nl`

[4] `http://www.mygrid.org.uk/usermanual1.7/scufl_language_wb_Features.html`

[5] `http://beanshell.org/`

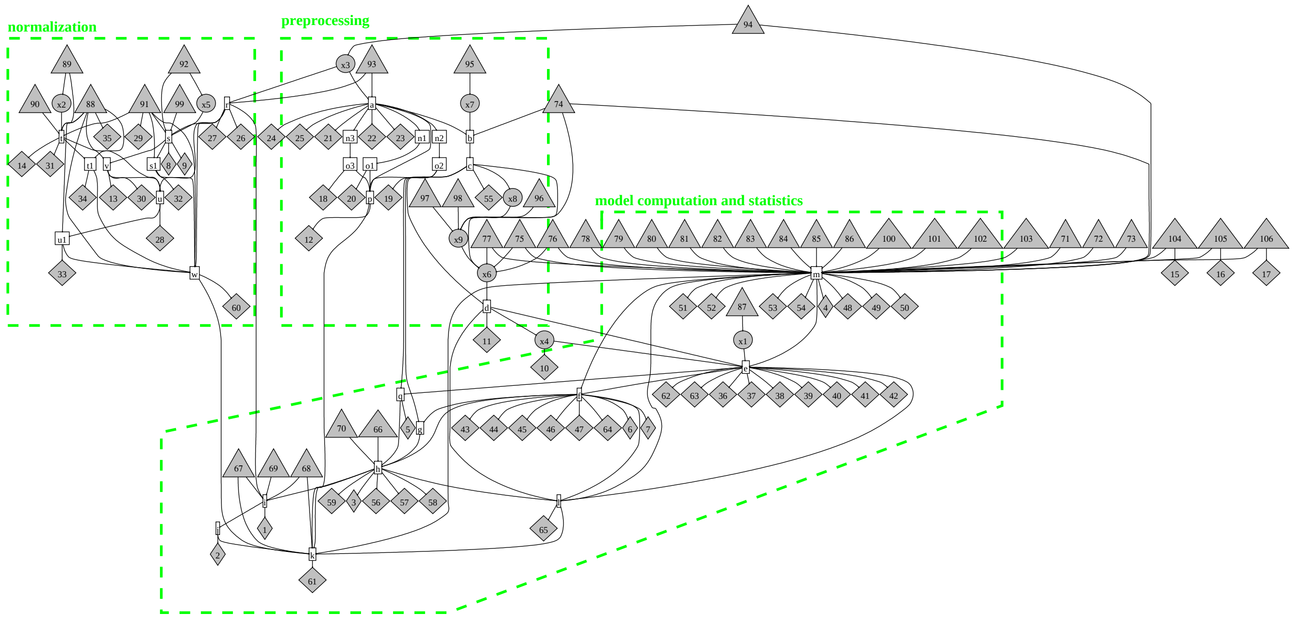


Figure 1: Workflow corresponding to the first-level analysis implemented by the *Feat* application from FSL. Triangles denote inputs and diamonds outputs. Boxes are processors executed on the grid and ellipses are local ones. For the sake of the legibility, we represented multiple data dependencies by single arrows, and processor ports and iteration strategies are omitted. Green dotted boxes indicate sub-workflows.

normalization of the fMRI scan (input 93[6]) to the standard brain (input 88) is composed of 3 registration steps. First, a high resolution anatomical scan of the subject (input 91) is independently mapped to the standard brain and to the fMRI scan. Then, both registration results are combined to produce the fMRI-to-standard-brain transformation. The inner sub-workflow deals with pre-processing of the fMRI scan, including motion and time correction, brain extraction and spatial filtering (a, b, c and d processors). The remaining sub-workflow corresponds to the computation of the model and activation statistics. They are then rendered (thresholded, clustered and overlaid on the fMRI scan) and an analysis report in HTML is finally generated. This workflow was executed on an analysis example involving all the implemented steps and validated against results obtained with the script implementation. We used a modified version of the Scufl "dot product" iteration strategy [6] to correctly describe job farming over several fMRI scans and sweeps over model parameters.

We chose to adopt a quite fine granularity to represent this workflow because it allows for more flexibility, since each component of the workflow (from the registration method to the results plotting format) can be customized. In a next step, granularity could be coarsened by gathering components in expandable sub-workflows.

Expressing a tool like *Feat* as a workflow is quite unwieldy, even when the underlying principle of the analysis is grasped. The application comes out of the FSL package as a 3,500 lines script, and its translation into a workflow representation is quite error-prone and difficult to validate. In the future, this kind of translation step could be avoided by directly developing such high-level applications as workflows from basic components of the software suite.

[6]The input numbers are given for the sake of completeness, since they are only well visible in the digital version of the paper

2.2 Discussion

A first glimpse at this workflow suggests that performance gains could be expected by exploiting intrinsic parallelism between sub-workflows. Sweeps over model parameters (inputs of processor `111`) may also save CPU time and data storage size with respect to the monolithic implementation by computing normalization and pre-processing only once. In Section 4 we discuss in more detail the potential performance gain based on a simulation study. Note that the *Feat* script also allows for selectively performing analysis steps; doing so for large experiments, however, involves some error-prone parameter settings, whereas a workflow engine manages it automatically.

Limitations remain though. Firstly, the management of workflow output proved to be more difficult than expected (see more details in Section 3). Secondly, the translation of the script into a workflow described in Scufl was not trivial. Some steps of the *Feat* first-level analysis have not been implemented in the workflow yet, for example B_0 unwarping, contrast masking, denoising and perfusion subtraction, which would increase the complexity and size of the workflow significantly. Additionally, we detected limitations in the expressiveness of the adopted workflow language to describe variable parameters. Generally speaking, “dynamic” workflow patterns are still difficult to describe in Scufl. Take the example of the number of EVs and contrasts, which have impact on model-related steps and the number of inputs and outputs. The number of input stimulus files (in case of custom-shaped stimuli signals) and the number of other parameter instances (such as the phase of the signal convolved to the stimulus) depend on the number of EVs. The cardinality of some other parameters may even depend on both the number of contrasts and EVs, such as for instance the weights given to the EVs in the contrast vectors. Ideally, all parts of the workflow should be dynamically adjusted to the number of contrasts and EVs. In practice, these are fixed parameters in the workflow, limiting its applicability to this particular fMRI experiment.

Lists manipulations could be envisaged to allow dynamic EVs and contrasts cardinality. Yet, if even possible, correctly assembling/splitting lists all along the workflow would burden the description a lot. This kind of problem is not specific to this particular fMRI use-case. It has been identified on several other applications and thus deserves thorough investigations.

Subtle changes in the experimental setup can also have significant impact on the workflow description. For instance, this implementation considers that each fMRI scan has one anatomical scan associated with it. One could wish to run experiments where a single anatomical scan is registered to all the fMRI scans of the same person. To be used for such a case, the workflow iteration strategies must be adapted.

3 Output organization

In order to be usable for real applications, a workflow system should not only be able to compute the results but also to deliver them in a suitable format. The workflow represented on figure 1 has 65 outputs that are stored as grid files for each run of the application. The workflow engine automatically stores the generated outputs as grid files with unique names, keeping track of provenance data. Even small parameter-sweeps or job farming experiments easily produce thousands of files. Depending on the nature of the underlying experiment, one could be interested only in a subset of those files that should be rapidly retrievable and accessible. Moreover, users may want to represent results from the same workflow in different ways. For example, one could focus on the influence of some parameter for a given contrast whereas the other would compare various contrasts for a given parameter set. An SPM user may also want to compare his/her results with output from FSL, thus requiring a different results organisation than a traditional FSL user.

Image analysis software has been addressing this problem for years by nicely organising output files in meaningful directory structures. For instance, the *Feat* application adopts the structure sketched in figure 2. Such a structure is semantically rich for the user: for example, checking the correctness of the registration procedure can be done easily by picking up files from directory `reg`. Summaries such as HTML reports can also be produced by the applications, thus helping to keep track of the various results produced by the execution (for instance `.../report.html`). However, when dealing with an experiment involving several runs of the application, it is then up to the scientist to organise the outputs in the most appropriate manner. Experienced users and other programs, however, may still require the *Feat* "native" output structure.

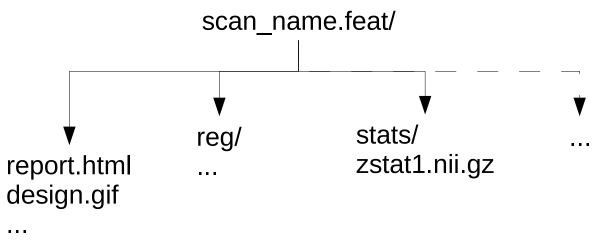


Figure 2: Directory structure adopted by the *Feat* script implementation to store results. Extensive description is available from the FSL website.

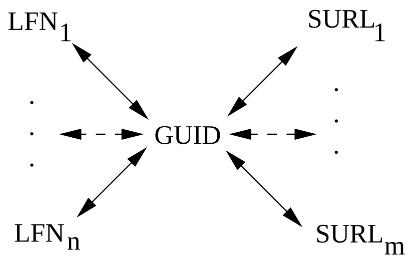


Figure 3: Classical data management architecture on production grids. A file is uniquely identified by a Grid Unique Identifier (GUID) to which n Logical File Names (LFN) may correspond in a file catalog (LFC). The file may be physically replicated on one or several sites, each of them providing a Site Unique Resource Locator (SURL).

Obtaining such a directory structure from grid workflow executions is not trivial. In this section we discuss alternatives to organize outputs of the workflow implementation. We assume that a workflow is made of *independent components* that can read/write files on a grid infrastructure. Files are supposed to be managed according to some grid middleware featuring (i) replica management among several grid sites and (ii) unified view of distributed storage through a unique file catalog. Figure 3 depicts the terminology used to refer to the various representations of a file, following the gLite middleware[7].

For instance a framework with such features could be a Scufl workflow made of Web-Services that submit jobs to the EGEE grid[8]. Each workflow component may be iterated several times during the execution, thus replicating part of the output directory structure. For instance, sweeping over a model parameter would generate several `stats` directories corresponding to a single `reg` directory. One could then choose either to create several `stats` directories in the same analysis directory (`analysis.feat/{reg,stats-1,stats-2,...}`) or to replicate the analysis directory for all the parameters (`analysis-1.feat/{reg,stats}, analysis-2.feat/{reg,stats},...`). Such choices should be made by the user. In any case, uniqueness of the file names produced by the workflow components should be guaranteed, which is difficult as components are independent from each other. Moreover, write permissions must be taken into account on shared environments.

[7]`http://glite.web.cern.ch`
[8]`http://www.eu-egee.org/`

3.1 Existing approaches to workflow output organization

The most convenient way for a workflow to store grid files is to use exclusively GUIDs. No name collision will occur among independent components and catalog permission problems are avoided. However, the file organization provided by GUID is completely meaningless and absolutely intractable for large numbers of files.

Information about the path of a result in the catalog could be set into the description of the workflow component that produces it. For instance, it could be specified that the `flirt` component of the *Feat* workflow always writes LFNs in $\texttt{\$dir}_i\texttt{/reg}$, where $\texttt{\$dir}_i$ is given as its input. This is the solution adopted in the GASW wrapping system that we used [5]. In practice, inconsistency problems rapidly arise from such a strategy and a global check of output paths is required. At some point, unique identifiers or workflow global variables (*e.g.* specifying the root directory of a workflow run) need to be generated. Besides, hard-coding the directory structure either in the component description or in the workflow representation itself is cumbersome and drastically limits reusability.

Another approach is to use workflow provenance to keep track of the relations between the produced data. The user is then able to navigate in a graph whose vertices are files and edges relations between them [13]. For the user the name of a file and its path in the result directory tree makes more sense than the list of the inputs that were used to produce it. Navigating across large data provenance graphs is also difficult.

A higher-level approach relies on results annotation and metadata browsing [11]. This better captures the semantic of an experiment than a directory structure. One could then retrieve files based on requests like ”Find anatomical-to-standard-brain transformations computed using X degrees of freedom”. However, in a grid environment, metadata consistency issues coming from file move, deletion or replication are difficult to handle.

None of the existing solutions completely satisfy the need to provide alternative and user-friendly ways to organize the files generated by a workflow.

4 Performance comparison of workflow and monolithic implementations

Comparing the performance of workflow and monolithic implementations can be envisaged on two different use-cases, job farming and parameter sweep. *Job farming* corresponds to the iteration of the complete workflow on a set of input data whereas *parameter sweep* corresponds to the iteration of the workflow on a range of parameters, given a single dataset. We assume that job farming replicates the whole workflow and parameter sweep only repeats parts of it. This holds in the FSL *Feat* workflow presented in Section 2.

The *makespan* of a workflow is the considered performance metric. It denotes the total elapsed time between the submission of the first job of the workflow and the completion of the last one. The number of computing resources (denoted n_R), the number of processed files (n_F) and/or parameters (n_P) are considered. In the following, we present some simulations with respect to those parameters, also studying the impact of data transfers and latency. They were conducted on the *Feat* FSL workflow presented in section 2 and performed by implementing a classical list scheduling algorithm. Resources are supposed to be homogeneous. Task execution times used in the simulations were measured on a Dual-Core AMD Opteron 2613.427 MHz with 3.5GB of RAM. Data transfer times have been measured between a cluster and an external Storage Element of the `vlemed` Virtual Organisation of the Dutch Virtual Laboratory for e-Science project[9].

[9]`http://www.vl-e.nl`

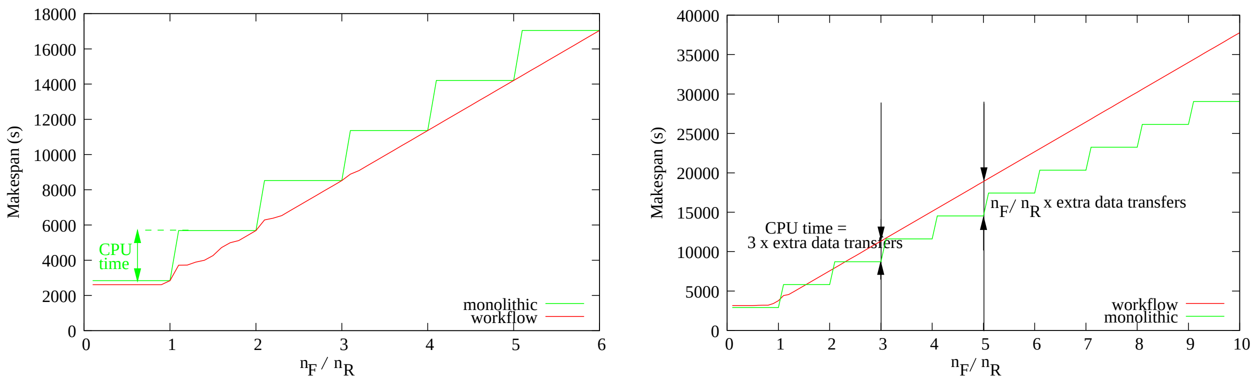


Figure 4: Simulation of the makespan of the *Feat* application for $n_R = 10$ resources. Left: ideal case. Right: including data transfers.

4.1 Simulations for the job-farming case

Ideal case. In the ideal case, data transfers are neglected and any free resource is supposed to be immediately available for computation (no latency). Under those assumptions, the workflow implementation cannot be less performing than the monolithic one, provided that a decent scheduling algorithm is implemented and neglecting the workflow management system overhead. Figure 4-left plots the evolution of the makespan of the *Feat* application with respect to the $\frac{n_F}{n_R}$ ratio for n_R=10 resources. The monolithic implementation exhibits a step shape. The height of each step is constant, and corresponds to the total CPU time to execute all the tasks in one run of the workflow (denoted T_{WF}). The workflow version reaches linear speed-up from $\frac{n_F}{n_R}$=3. Both implementations perform the same for integer fractions $\frac{n_F}{n_R}$, in which case an optimal usage of resources is guaranteed. Apart from those particular cases, the workflow version clearly outperforms the monolithic one. Best gains are achieved for $\frac{n_F+1}{n_R}$ and are upper-bounded by T_{WF}. This gain remains constant with respect to n_F. This performance improvement comes from the exploitation of the intrinsic parallelism in the workflow, in particular due to the registration steps. Note that finer granularity of the tasks also allows better scheduling.

Impact of the data transfers. Figure 4-right plots the evolution of the makespan of the *Feat* application taking into account data transfers and for n_R=10. This simulation has been obtained by increasing workflow tasks sizes by the duration of the data transfers they require, which assumes that data transfers are homogeneous among the grid nodes. Although not completely realistic, this assumption can be reasonable, in particular when considering production grids, where input/output data has to be fetched/registered from external data storage servers. Both the workflow and the monolithic implementations exhibit comparable performance for $\frac{n_F}{n_R}$ in [0,3]. Beyond this interval, the monolithic version performs better. Extra data transfers coming from the workflow implementation can be visualised on the graph at integer values of $\frac{n_F}{n_R}$. In particular, we can notice at $\frac{n_F}{n_R} = 3$ that the overhead coming from those extra data transfers is about a third of T_{WF}.

Latency and data transfers. Here latency is defined as the time needed to access a free CPU on the infrastructure. It includes for instance the submission and scheduling time, but excludes queueing time in case of a loaded infrastructure. Figure 5-left depicts the comparison of monolithic and workflow implementations taking into account both data transfers and various values of the latency and for $n_R = 10$ resources. As it

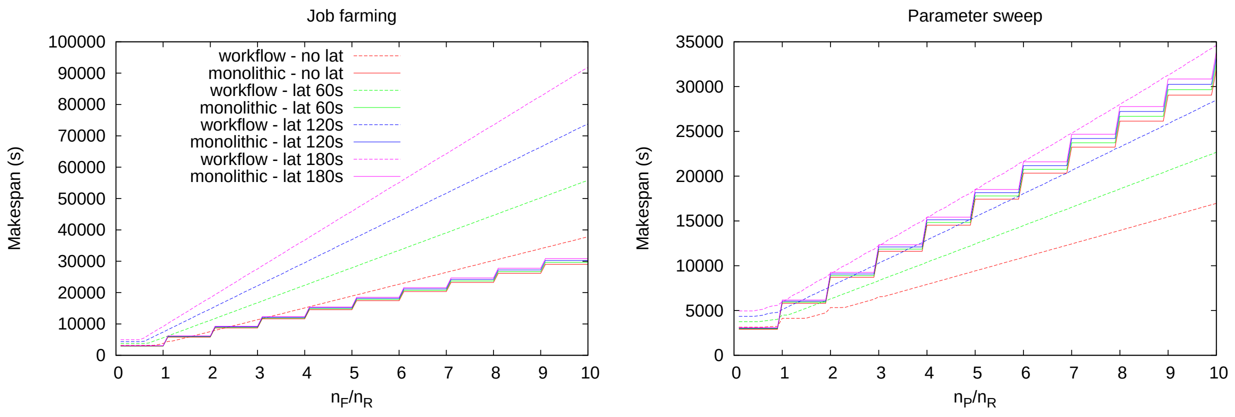


Figure 5: Simulation of the makespan of the *Feat* application, taking into account data transfers, for latency ranging from 0s to 180s. Colours represent different values of the latency. Left: job farming; workflow performance is crippled by data transfers and latency. Right: sweep over a model parameter; the workflow version outperforms the monolithic one for latency values lower than 3 minutes.

could be expected, the impact of the latency is much more dramatic on the workflow implementation than on the monolithic one. For a 1-minute latency (green curves), the performance of the workflow implementation is still comparable to the monolithic one for $\frac{n_F}{n_R}$ in [0,1] while this range is [0,3] without latency (red curves). Then, the monolithic version rapidly outperforms the workflow, in particular at higher latency values (blue and pink curves). As already mentioned, we chose to adopt a quite fine granularity in this workflow, for reusability reasons. Coarser granularity would limit the impact of latency and data transfers, for example, by job grouping strategies as adopted in [5]. Yet, powerful strategies are still challenging to elaborate in the general case.

Increasing CPU performance. Increasing average CPU performance of the infrastructure certainly will affect the gain yielded by the workflow implementation. It can be noticed from left of figure 4 that a reduction of the CPU time T_{WF} will decrease the height of the steps in the monolithic curve, thus reducing the gap to the workflow red curve. Similar conclusion can be made when data transfers are taken into account: in this case, their impact would be increased by a improvement of the average CPU performance of the infrastructure.

4.2 Simulation for the parameter sweep case

Figure 5-right plots a simulation of the makespan of the application for the parameter sweep use-case for n_R=10 resources. It takes into account data transfers and represents the makespan for several different latency values (denoted by the different colours). In spite of the data transfers, the workflow implementation clearly surpasses the monolithic one when resources are accessible without any latency (red curves). Besides, this gain increases with respect to $\frac{n_P}{n_R}$. The performance of the workflow implementation then gets closer to the monolithic one when the latency value increases (green and blue curves). When the latency reaches 3 minutes (pink curves), the monolithic version starts to outperform the workflow one for all values of $\frac{n_P}{n_R}$.

4.3 Discussion

The fact that grid latency impacts job-farming much more dramatically than parameter-sweep comes from differences in the number of grid jobs implied in those use-cases. Adding a file to a job farming experiment (*i.e.* incrementing n_F) leads to the submission of 30 additional grid jobs (1 per workflow processor) whereas adding a parameter to a sweep experiment (*i.e.* incrementing n_P) only generates 10 jobs (1 per processor of the model computation sub-workflow on figure 1).

Figure 5 gives an idea about how the monolithic and the workflow implementations would compare on different kind of infrastructures. Red curves might correspond to clusters or small scale grids, and green to pink curves would denote grids with increasing sizes or loads. In practice, we commonly observe a one to two minutes latency on the infrastructure of the Dutch VL-e project. Thus, this simulation suggests that, in most practical cases, sweeping over a model parameter would benefit from a workflow implementation.

The simulation results presented here were computed only for n_R=10 resources. A formal proof is out of the scope of this paper, however we can show that only the $\frac{n_F}{n_R}$ ratio (or $\frac{n_P}{n_R}$ for parameter sweeps) matters to the performance. Moreover, we can state that once the workflow makespan has reached linear behavior (as it is the case on figures 4 to 5), it will no longer deviate from it for larger values of $\frac{n_F}{n_R}$.

5 Conclusions

We presented a grid workflow implementing the *Feat* application of the FSL package. The description of this quite complex application was possible using a state-of-the-art workflow language, Scufl. The workflow implementation was validated on the grid infrastructure of the Dutch VL-e project.

This exercise was important to identify the pros-and-cons of workflow implementation for complex (image analysis) applications. Contrary to the expectations, simulations reveal that the performance improvement yielded by the workflow implementation in a job farming scenario is limited. Only a fixed gain is obtained on ideal grid infrastructures, and in practice, data transfers and latency rapidly take it down. Reducing the granularity of the workflow, for example by grouping of components, could be beneficial, however it is difficult to achieve in the general case. Conclusions are quite different when considering a model parameter sweep experiment, however. Despite the data transfers, the workflow implementation outperforms the monolithic script, even for large latencies.

This study also identified additional problems to be tackled. Firstly, the generality of this implementation is still limited. Various *Feat* use-cases require ad-hoc workflow modifications to implement changes *e.g.* in the number of contrasts or in the experimental set-up. This workflow rapidly produces thousands of files for a simple experiment, which makes intuitive output organization a challenging problem. Hard-coding a directory structure in workflow components is feasible, but it drastically limits the reusability of both the workflow and its components. We are currently investigating how to map data-level workflow provenance to a grid file catalog directory structure that is more intuitive to the end-user. Lessons learned from this study lead us to state that the results structure specification should be independent (i) from the workflow components, (ii) from the workflow description and (iii) from the workflow execution.

Thus, using workflows for such a widely scoped application is still challenging in terms of description, usability and performance. Hard-coded ad-hoc solutions can yield good benefits for some specific use-cases but further research to improve generality and reusability has to be considered. All in all, well designed (monolithic) applications such as FSL Feat are still hard to beat with generic workflow management systems.

Expected benefits of workflow management on the work style of neuroscientists make this challenge worth considering though. Apart from the performance and usability items mentioned in this paper, one could forecast advantages ranging from data and knowledge management to experiment setup and execution. Data produced by a workflow could be automatically annotated, enabling queries such as ”find any experiment conducted on brain region X considering fMRI task Y”. Moreover, once analysis packages are described as workflows, customizing them to particular use-cases becomes accessible for end-users, who will then be able to conduct analysis experiments more autonomously.

6 Acknowledgements

This work was carried out in the context of the VL-e project, which is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W) and is part of the ICT innovation program of the Ministry of Economic Affairs (EZ). We acknowledge Marie-José van Tol, Remi Soleman and Aart Nederveen for providing concrete large-scale detailed use-cases of fMRI analysis motivating this work. We also thank Kamel Boulebiar for smart technical developments related to gLite data management.

References

[1] R. Andrade et al. A Grid Framework for Non-Linear Brain fMRI Analysis. In *HealthGrid*, pages 299–305, Geneva, 2007.

[2] D. Churches et al. A Parallel Implementation of the Inspiral Search Algorithm using Triana. In *Proceedings of the UK e-Science All Hands Meeting*, Nottingham, UK, Sept. 2003.

[3] E. Deelman et al. Mapping Abstract Complex Workflows onto Grid Environments. *Journal of Grid Computing*, 1(1):9–23, 2003.

[4] K. Friston. *Statistical parametric mapping and other analysis of functional imaging data*, pages 363–385. AcademicPress, 1996.

[5] T. Glatard et al. A Service-Oriented Architecture enabling dynamic services grouping for optimizing distributed workflows execution. *Future Generation Computer Systems*, 24(7):720–730, 2008.

[6] T. Glatard et al. Flexible and efficient workflow deployement of data-intensive applications on grids with MOTEUR. *International Journal of High Performance Computing and Applications (IJHPCA)*, 22(2), 2008.

[7] R. Goebel. *BrainVoyager: Ein Programm zur Analyse und Visualisierung von Magnetresonanztomographiedaten*. Göttingen: Gesellschaft für wissenscha, 1997.

[8] T. Oinn et al. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics journal*, 17(20):3045–3054, 2004.

[9] S. Olabarriaga et al. Parameter Sweeps for Functional MRI Research in the Virtual Laboratory for e-Science Project. In *Workshop on Biomedical Computations on the Grid (Biogrid'07)*, pages 685–690, Rio de Janeiro, May 2007.

[10] D. Rex et al. The LONI Pipeline Processing Environment. *NeuroImage*, 3(19):1033–1048, July 2003.

[11] N. Santos et al. Distributed Metadata with the AMGA Metadata Catalogue. In *Workshop on Next-Generation Distributed Data Management (HPDC'06)*, Paris, France, June 2006.

[12] S. Smith et al. Advances in functional and structural MR image analysis and implementation as FSL. *NeuroImage*, 23(1):S208–S219, 2004.

[13] J. Zhao et al. Using Semantic Web Technologies for Representing e-Science Provenance. In *Third International Semantic Web Conference (ISWC2004)*, pages 92–106, Hiroshima, Nov. 2004.

The Application of a Service-Oriented Infrastructure to Support Medical Research in Mammography

Christopher Tromans[1], Sir Michael Brady[1], David Power[2],
Mark Slaymaker[2], Douglas Russell[2] and Andrew Simpson[2]

[1]Wolfson Medical Vision Laboratory, Department of Engineering Science, University of Oxford, Parks Road, Oxford, UK, OX1 3PJ.
[2]Computing Laboratory, University of Oxford, Parks Road, Oxford, UK, OX1 3QD.

Abstract

We describe the application of a framework for secure access to, and sharing of, data from disparate data sources. The approach ensures that data is shared only in circumstances permitted by the data owner. In a healthcare context, such circumstances are often limited by legal and ethical concerns. Moreover, our approach allows for the aggregation of data from disparate data sources. The approach offers the potential to contribute to the ideal of fully connected healthcare. The specific use case of Digital Mammography is discussed.

Contents

1 Measuring Radiodensity in Mammographic Images 44

2 Middleware Architecture 46

3 Connectivity within the Clinical Trial 49

4 Discussion 51

Since May 2006, the authors (and other colleagues) have been working on a collaborative research project, GIMI (Generic Infrastructure for Medical Informatics), which aims to develop a secure computer infrastructure to support medical applications. The project consists of a middleware development team as well as three application teams, with the focus of the middleware being the facilitation of secure and ethical aggregation of distributed data from remote sources. The middleware must be interoperable in the sense that it is able to interface with the wide diversity of existing (and future) health information systems in place around the world, and to provide both secure access and secure transfer of data. The middleware is based on the principles of [1], and its implementation is termed *sif* (service-oriented interoperability framework) [2].

The application teams are contributing to the validation of *sif* through the input of requirements and evaluation of prototypes. An iterative development cycle is helping to ensure that the framework meets the needs of a wide variety of applications. Within the project, the applications are: the use of the technology in studying patients suffering from long-term conditions such as diabetes and asthma; mammography training and auditing; and image analysis for cancer research. This paper describes the latter application, and particular attention is given to the analysis of mammographic images. Although this paper focuses on a particular application, we stress that the *sif* architecture is generic in that it can expose data from a wide variety of data sources.

Medical research, in particular studies of an epidemiological nature, or those in which an algorithm needs to learn from a sufficiently comprehensive training set, require large datasets in order that sufficient statistical power be attained, that is it should include a comprehensive sampling of the total, clinically significant variation present within the population. Since it is extremely rare that a single institution can provide such a comprehensive sample, such studies typically necessitate multiple institutions, spread sparsely geographically. (A particular example of this is a phase 3 clinical trial of a drug, or PMA testing of a new device or software system.) In such circumstances, an efficient and secure data transfer infrastructure is essential if data is to be shared effectively in real time (which, for example, it is typically not in the case of a phase 3 clinical trial). In this paper we describe such a study we are preparing to conduct for the evaluation of a technique we have been developing over the past five years for measuring radiodensity in mammographic images. The image processing algorithm is first to be assessed through synthetic phantom experimentation: the imaging of test objects for which the radiodensity is known precisely, and thus the accuracy of the technique in describing reality may be established. When the experimental results deem the performance of the normalisation software to be acceptable a study will commence with our clinical partner on a 1.25 terabyte patient dataset [3]. The dataset consists of all the digital mammographic images acquired over a two year period at their clinic (using an IMS Giotto full field digital mammography unit), and it is hoped that these will all be processed and subsequently analysed. Analysis will aim to identify any relationship between the observed normalised radiodensity measures and the underlying pathologies of both tumour and healthy tissue, as well as the assessment of risk as currently considered in breast density. Currently, radiologists identify possibly malignant lesions through “contrast features”, for example dense lines emanating from a density (termed spicules) suggesting invasion of the surrounding tissue by a malignancy, or small bright round densities, which are possible microcalcifications. The use of radiodensity adds yet another weapon into the armoury to verify that the feature identified is in fact that which is suspected. For example, in the case of a microcalcification, a measurement of the radiodensity in the small region of interest, relative to its surroundings, can be checked to ensure the feature is exhibiting the likely attenuation properties of calcium. This paper describes the use of *sif* to support the continued development of this algorithm, and, in particular, its clinical assessment and validation.

1 Measuring Radiodensity in Mammographic Images

For several decades the correlation between radiological features of the breast and the likelihood of the breast containing, or subsequently developing, a malignant lesion, has been studied in the field of research termed breast density. Work in this area was pioneered by Wolfe in 1969 [4] who proposed a four category classification for assessing mammographic parenchymal patterns: in particular the prominence of ductal patterns and the occurrence of connective tissue hyperplasia. Wolfe presented findings showing that each of the four groups, from lowest to highest density, had breast cancer incidence rates of 0.1, 0.4, 1.7 and 2.2 [5]. Since Wolfe's pioneering studies, much work has been contributed to the field by many different authors. However, the most sustained and significant of these contributions is that made by Norman Boyd and his colleagues. In 1982 Boyd et al [6] defined a six category classification (SCC) which focuses on mammographic hyperplasia, and through the use of this assessment technique reported good results in discriminating between images on the basis of their likelihood of acquiring breast cancer. However, these measures suffer a shared limitation, namely reader subjectivity: one highly experienced radiologist may place an image in one category, whilst another may argue that it falls into a different one. This led Byng et al [7] to develop an interactive thresholding technique to segment, and thereby quantify, mammographic hyperplasia. Such measures are termed "area measurements" since they treat the mammogram as a 2D image, ignoring the third dimension (depth through the breast), and treat the projected image as entirely representative. In fact, mammograms are integrated x-ray attenuation maps of the breast, which is tightly compressed to spread dense tissue patches that may be indicative of cancer. A mammogram results from projecting the real 3D compressed breast to a 2D image, thereby losing all depth information between the plates.

To take account of the three-dimensional breast, "volumetric measurements" of breast density have been developed. Such measures quantify the tissue present in the cone between a detector pixel, and the x-ray focal spot, using the likely x-ray attenuation coefficients of the constituent tissues. In 1996 Highnam and Brady proposed [8] the h_{int} representation which measures volumetric density. They developed a model of image formation considering the path of x-ray photons from point of emission in the x-ray tube, to absorption at the detector. Several alternative techniques of measurement have been subsequently proposed, for example Kaufhold et al [9], whom approximate a transfer function describing imaging formation gleaned from tissue equivalent phantom images.

Inspired by Highnam and Brady's work [10], we have developed a second generation of their model [3, 11, 12]. The extra power made available by modern computers, and advances in medical physics, has enabled the removal of many of the simplifying assumptions in their model, leading to a more comprehensive image formation model. The developed algorithm consists of an image formation model considering the production of x-ray photons in the tube, their interactions through absorption and scattering phenomena within the breast, the selective absorption process occurring dependant on angle of incidence as they pass an anti-scatter grid, and their subsequent detection at the image receptor pixel. This model is used to calculate the equivalent thickness of reference material required to create the observed intensity at each pixel within the image, independent of scattered radiation. This thickness, divided by the ray traversal distance, yields the normalised radiodensity measure, which is thus independent of the incident photon flux characteristics, the detector response, and the breast thickness. To overcome the limitations in anatomical representation we highlighted in [12] an alternative normalised measure has been adopted in which the attenuation of the breast per unit traversal distance is compared to that of a reference material. This is analogous to the universally used Hounsfield unit (HU) in Computed Tomography (CT). Figure 1 shows one of the tissue equivalent test objects used for validation in a partially assembled state, and the clinically installed IMS Giotto full field digital mammography unit.

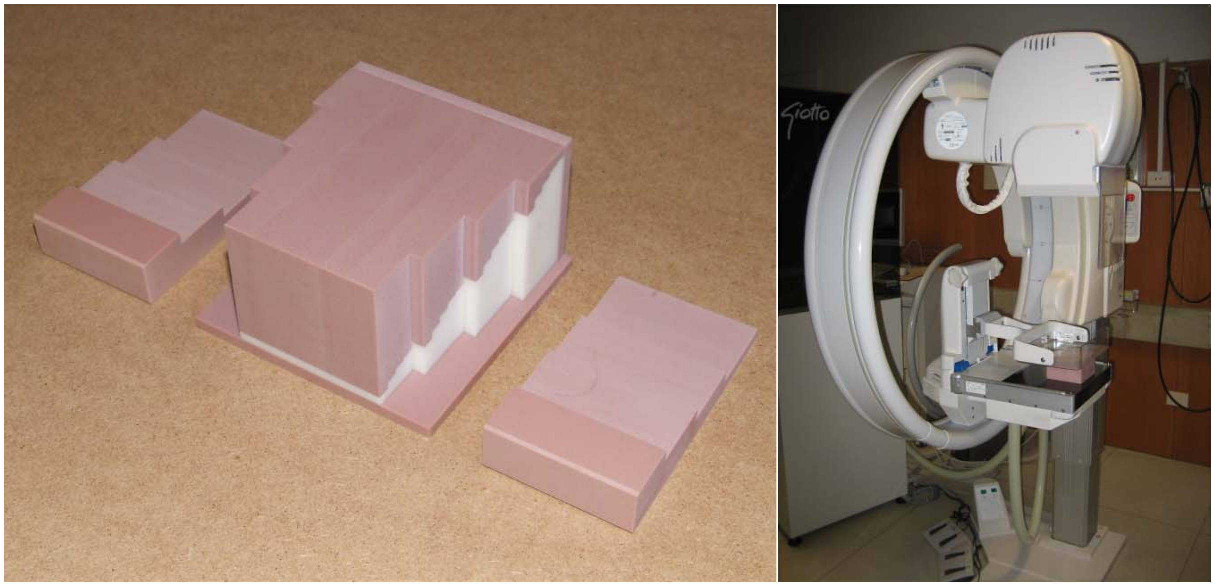


Figure 1 The mammographic test object manufactured from tissue-equivalent resins used for algorithm validation, and the IMS Giotto MD.

2 The *sif* Architecture

The *sif* architecture was developed to fulfil the need to share distributed data in a secure, federated and data-agnostic fashion. Its design was influenced by the eDiaMoND [13] and MammoGrid [14] projects, both of which the first two authors were involved in.

Following [1], *sif* has a clear requirement to facilitate fine-grained access control to data sources, with policies being determined locally by data owners; there is also a need to guarantee secure transfer between endpoints. A key driver has been the joining of data sources to provide “bigger and better” research and healthcare: fundamental to this is the presentation of a federation of multiple data sources as a single entity as described in [15]. In order to develop a generic solution we have taken a data-agnostic approach, by which we mean that *sif* is capable of “plugging in” to a variety of data sources regardless of underlying technology or data format. (See [2] for an overview of the motivation for sif.)
It is, perhaps, worth reflecting upon the philosophy behind, and the current status of, the middleware.

Some of the requirements that have informed the design and development are described below.

- *Facilitating buy-in*: In the ‘real world’, complexity and cost are factors that significantly influence procurement decisions. Yet, they are often not mentioned in the health grid context. It has been important to us that researchers should be able to utilise data sources---without requiring them to be ported to new operating systems or database management systems, or requiring the data to be transformed into a new structure to facilitate interoperability. This is contrary to the approach adopted in the eDiamond and Mammogrid projects, and the “Grid image workflow paradigm” followed by Globus MEDICUS [16], where fixed data schemas have been designed and implemented within which the DICOM data is stored upon the grid nodes themselves. It has also been important to us that application developers should be able to ‘pick up and play with’ sif: to this end, data is accessed through a simple API.

- *Abstraction*: Through experience, we have come to the conclusion that it is only through a loosely-coupled approach that the delivery of more genuinely generic solutions are possible. Thus, one of our drivers is the facilitation of technology-agnosticism for application developers via the data-agnosticism philosophy of *sif*.

- *Interoperability*: If one were to take a simplified view, one might characterise the issue of data interoperability as facilitating both *database interoperability* (between Dr Smith's breast cancer research database in San Francisco and Dr Thomas' colorectal cancer research database in New York) and *database management system interoperability* (between the IBM DB2 database utilised by Dr Smith and the Oracle database utilised by Dr Thomas). Our concern is the latter; the issue of what we might term semantic interoperability is left to application developers. This leads to a 'bottom-up'---as opposed to a 'top-down'---construction of virtual organisations, which we shall explore in the next section.

- *Security*: In a health grid context, one can think about security in terms of storage, access and transfer. With respect to a *sif* deployment, the responsibility for secure storage resides with the data owner; as such this is not a concern here. Secure access and transfer, are, however, essential concerns. With respect to the former, access policies are constructed by data owners; with respect to the latter, secure channels are established between external nodes. The requirements for secure access and transfer have been influence by relevant UK and European legislation.

At the heart of *sif* is a plugin mechanism, through which the goal of interoperability across diverse systems is realised. Each plugin is capable of connecting to a particular resource, presenting a standardised interface to the middleware, which in turn provides a federated view to applications. All communications between the resource and the client pass through the middleware using the standard interfaces provided. The middleware thus provides a decoupling between client and resource: that is, the exact details of the resource are abstracted away from the client. The middleware, acting as an intermediary, provides encrypted client connections to ensure secure data transfer across public and private networks, as well as client user identification and authorisation. The identification mechanism utilises secure signed certificates, and once authenticated, a user is subject to the data access restrictions set by the data owner.

There are three types of plugin:

- *Data plugins* are concerned with interfacing queryable data sources to the middleware. Examples of such data sources include relational databases and XML databases.

- *File plugins* are concerned with interfacing file systems, be they local file systems, or network file systems (such as NFS or CIFS).

- *Algorithm plugins* are concerned with the facilitation of the (possible remote) execution of algorithms on data. Examples of algorithms include basic image processing or reconstruction routines.

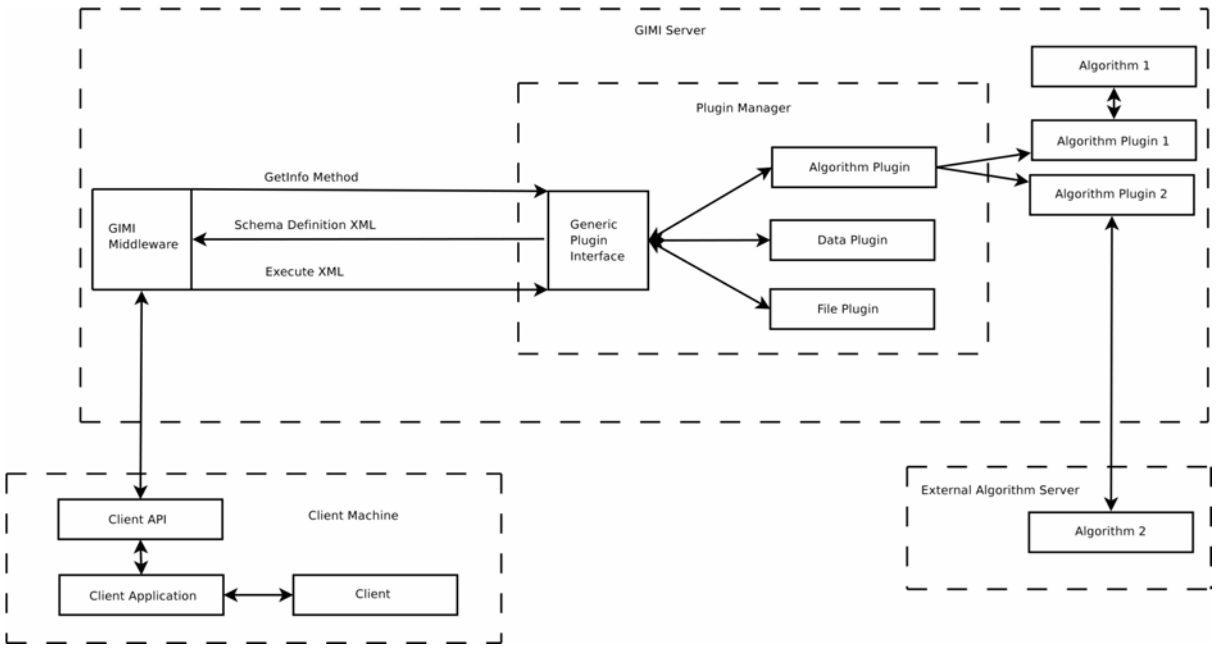


Figure 2 An overview of the plugin architecture.

Through the adoption of a standard interface for each of the three types of plugin it becomes possible to add heterogeneous resources into the infrastructure. These can be managed by the application developer who has the ability to install, uninstall, and update plugins. Importantly, there is no need for the resource being advertised through the plugin system to directly represent the physical resource. What is advertised as a single data source may come from any number of physical resources, or even another distributed system.

Figure 2 shows the overall architecture of the plugin system, with particular focus on two algorithm plugins. A client is interacting with an application running on their local workstation. The application communicates with the server via the sif client API, which is standard for all applications. The application uses the *GetInfo* method to obtain the definition of the algorithm input and outputs which are represented as XML schemas. The client provides appropriate inputs which are then passed through the *sif* API to the server as an XML document. The requested algorithm plugin is then executed with the given inputs contained within the XML document. In the example diagram, Algorithm 1 is executed locally and Algorithm 2 is executed on a remote machine. In the case of Algorithm 2 the job of the plugin is to communicate with the external algorithm server. The results of the execution (which must be consistent with the output schema) are returned to the client application which then handles them as appropriate.

In a distributed context, *sif* acts as a federation layer: if a user runs a query across several data nodes, then the middleware will distribute that query to the nodes and aggregate the results. However, the reason that sif can expose a wide variety of data sources is that it makes no assumptions about structure or semantics: while sif facilitates distributed queries, it is up to the end-user (or application) to ensure that the queries (and results) are meaningful. This, of course, makes the task of federation much easier. Other issues, such as guaranteed consistency and replication are also irrelevant in this context: the former because access is read-only; the latter because that is the responsibility of the data owner---each site is considered autonomous.

Figure 3 depicts the operation of the federation functionality. The client first formulates a federated query, which is made up of sub-queries and details of how their results should be combined. Each sub-query contains details of the server which hosts the plugin, the plugin identifier and the query to be executed. In step 1 the client application utilising the API sends a query to the client's local node. The local node then decomposes the overall query into its individual query elements. These individual queries are then forwarded to the relevant remote nodes (step 2), or processed locally, if appropriate. Each node then processes each individual query it has received before returning (step 3) the results along with additional information about the success of the processing to the originating local node. Once a result has been received by the local node from all the nodes processing queries, the resulting data is then combined as requested by the original query submitted by the client. In addition, all the information relating to the success (or otherwise) of each of the sub queries is aggregated. This combined data and report on the processing is then returned to the client application (step 4).

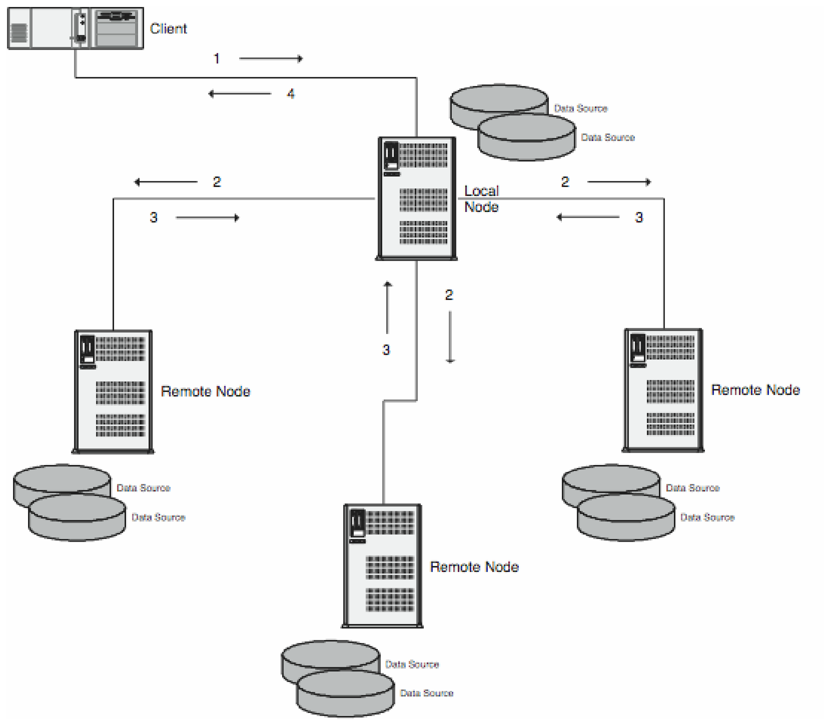


Figure 3 A federated query

3 Connectivity within the Clinical Trial

As alluded to earlier, there are, within the GIMI project, three application areas that are helping to validate the development of *sif*. In this section we describe one of these applications, which pertains to the analysis of mammographic images. In particular, the application is concerned with the validation and continued development of the algorithm described in Section 1.

The Giotto Image MD, like most digital mammography systems, incorporates DICOM 3.0 communication standards [17] allowing it to fit into any RIS/PACS environment. Our clinical partner has the unit connected to a full PACS system to which all acquired images are sent. As well as the image, the PACS system holds the associated metadata, some of which is patient sensitive, names, dates of birth, identification numbers and such.

The short-term goal driving the application of *sif* that is described in this paper is to facilitate direct access to image data from the PACS server of our clinical partner, by overlaying it with both a file (for transfer of image data) and data plugin (for querying the metadata to establish which studies to request via the file plugin). So, for example, in the event of a phantom acquisition being required, this could be acquired by a local radiographer, sent to the PACS system, and then be available to a researcher immediately—in contrast to the current arrangement of writing recordable media, such as a DVD, and sending it via the public post (which raises issues of integrity and reliability). In the longer term, we would hope to use a similar approach to gain access to anonymised patient images and associated metadata.

Of course, consideration must be given to protecting patient privacy and confidentiality: simply adding an AET (application entity title) for the researcher is not appropriate, since this only allows very coarse restrictions on the basis of IP address and TCP port. The approach to access control provided by *sif*—whereby fine-grained authorisation policies can be constructed locally by data owners in accordance with their needs (and, in some cases, obligations)—has the potential to provide assurance to these data owners that any access granted will be restricted appropriately. This, then, has the potential to offer functionality over and above that afforded by the all-or-nothing approach of the addition of an AET.

Currently, we are utilising file, data and algorithm plugins to refine the existing algorithm on legacy data sets. Two DICOM operations are of primary interest in this respect:

- **C-MOVE.** This operation is used to move image data. The C-MOVE SCP (Service Class Provider) is requested to act as a C-STORE SCU (Service Class User) and to copy composite instances to a requested AET, which may or may not be the original C-MOVE SCU (although it normally is). Interfacing to this operation has been successfully implemented as a file plugin.

- **C-FIND.** This operation is akin to an SQL query, whereby a data set is passed from the SCU to the SCP containing two sorts of attribute:

    - those which need to be matched (equivalent to the WHERE clause of an SQL query); and

    - those to be returned to the SCU (equivalent to the SELECT clause of an SQL query).

  Interfacing to this operation has been successfully implemented as a data plugin.

The algorithm plugin facility allows us to share our image processing and normalisation algorithms with our clinical partners. Figure 4 shows the output of the segmentation algorithm used as an input to the normalisation routines, which has been shared in this way. Efficient bidirectional communication is thus possible which provides an iterative development cycle in which new versions of an algorithm may be made available instantly to the clinical users, who are then able to apply it to the available images, review the output, and subsequently report back the success or limitations of the refinements. Complications of keeping many different potentially remote computers up-to-date are thus reduced. In addition, it facilitates access to an algorithm without the need to distribute the code or a binary executable (which might be reverse-engineered): this, then, has the potential to enable collaboration without risking the loss or compromise of intellectual property.

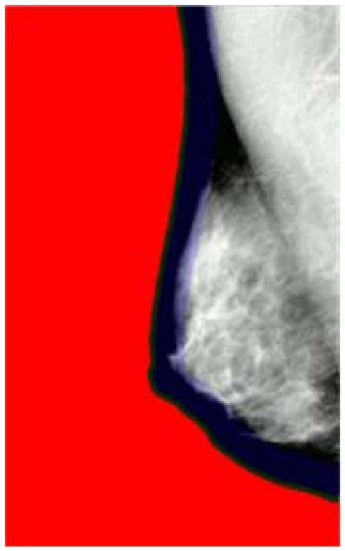


Figure 4 The output of the algorithm to segment the breast-air boundary and inner periphery at which the breast begins to occupy the full separation between the compression paddles.

4 Discussion

The vast upsurge in the popularity of digital mammography, and in digital imaging in general, has led to a need for secure and scalable computing infrastructures that can provide secure and ethical access to remote and distributed data to those who have a legitimate usage requirement. The middleware described in this paper, though it continues to be under development, shows considerable promise in fulfilling this need. The development of *sif* within the GIMI project is being undertaken in collaboration with a number of application teams, drawn from a variety of domains. This close user involvement is ensuring that the framework developed meets the needs of a diverse community. In this paper, we have concentrated on one such application, pertaining to image analysis of mammograms. Currently, this application of *sif* is helping to validate and refine a normalisation algorithm; in the longer term the framework has the potential to facilitate genuine, real-time "joined-up" healthcare.

Reference

[1] A. Simpson, D. Power, M. Slaymaker, and E. A. Politou, "GIMI: generic infrastructure for medical informatics," presented at 18th IEEE Symposium on Computer-Based Medical Systems Proceedings, 2005., 2005.

[2] A. C. Simpson, D. J. Power, D. Russell, M. A. Slaymaker, G. K. Mostefaoui, G. Wilson, and X. Ma, "The development, testing, and deployment of a web services infrastructure for distributed healthcare delivery, research, and training," in *Managing Web Services Quality: Measuring Outcomes and Effectiveness (to appear)*, 2008.

[3] C. Tromans, M. Brady, D. Van de Sompel, M. Lorenzon, M. Bazzocchi, and C. Zuiani, "Progress Toward a Quantitative Scale for Describing Radiodensity in Mammographic Images," presented at International Workshop on Digital Mammography, 2008.

[4] J. N. Wolfe, "The prominent duct pattern as an indicator of cancer risk," *Oncology*, vol. 23, pp. 149-58, 1969.

[5] J. N. Wolfe, "Risk for breast cancer development determined by mammographic parenchymal pattern," *Cancer*, vol. 37, pp. 2486-92, 1976.

[6] N. F. Boyd, B. O'Sullivan, J. E. Campbell, E. Fishell, I. Simor, G. Cooke, and T. Germanson, "Mammographic signs as risk factors for breast cancer," *Br J Cancer*, vol. 45, pp. 185-93, 1982.

[7] J. W. Byng, N. F. Boyd, E. Fishell, R. A. Jong, and M. J. Yaffe, "The quantitative analysis of mammographic densities," *Phys Med Biol*, vol. 39, pp. 1629-38, 1994.

[8] R. Highnam, M. Brady, and B. Shepstone, "A representation for mammographic image processing," *Med Image Anal*, vol. 1, pp. 1-18, 1996.

[9] J. Kaufhold, J. A. Thomas, J. W. Eberhard, C. E. Galbo, and D. E. Trotter, "A calibration approach to glandular tissue composition estimation in digital mammography," *Med Phys*, vol. 29, pp. 1867-80, 2002.

[10] R. Highnam and M. Brady, *Mammographic image analysis*. Dordrecht; London: Kluwer Academic, 1999.

[11] C. Tromans, "DPhil Thesis: Measuring Breast Density from X-Ray Mammograms," in *Engineering Science*: Oxford University, October 2006.

[12] C. Tromans and M. Brady, "An Alternative Approach to Measuring Volumetric Mammographic Breast Density," presented at International Workshop on Digital Mammography, 2006.

[13] J. M. Brady, D. J. Gavaghan, A. C. Simpson, M. Mulet-Parada, and R. P. Highnam, *eDiaMoND: A Grid-enabled federated database of annotated mammograms*: Wiley, 2003.

[14] S. R. Amendolia, F. Estrella, W. Hassan, T. Hauer, D. Manset, R. McClatchey, D. Rogulin, and T. Solomonides, "MammoGrid: A Service Oriented Architecture Based Medical Grid Application," *Lecture Notes in Computer Science*, vol. 3251, pp. 939-942, 2004.

[15] M. A. Slaymaker, D. J. Power, D. Russell, G. Wilson, and A. C. Simpson, "Accessing and aggregating legacy data sources for healthcare research, delivery and training," presented at SAC, 2008.

[16] S. G. Erberich, J. C. Silverstein, A. Chervenak, R. Schuler, M. D. Nelson, and C. Kesselman, "Globus MEDICUS - Federation of DICOM Medical Imaging Devices into Healthcare Grids," presented at HealthGrid, 2007.

[17] NEMA, *Digital Imaging and Communications in Medicine (DICOM) Standard, version 3*, 2000.

Healthgrids, From Revolution to Evolution - Experiences from the Health-e-Child and neuGRID European Projects

David Manset

Director of Biomedical Applications
maatGknowledge, ES
France
dmanset@maat-g.com

Invited talk

Abstract

Based on a recent survey of ongoing eHealth European projects, this presentation will draw some interesting conclusions on the difficulties faced in the adoption of Grid technologies. These conclusions will be supplemented with concrete results from the ongoing Health-e-Child and neuGRID projects.

- Health-e-Child project: http://www.health-e-child.org/
- neuGrid projet: http://www.neugrid.eu/

Multiple Sclerosis Brain MRI Segmentation Workflow deployment on the EGEE Grid

Erik Pernod[1], Jean-Christophe Souplet[1], Javier Rojas Balderrama[2], Diane Lingrand[2] and Xavier Pennec[1]

[1]INRIA, Asclépios Project team, 2004 Route des Lucioles BP 93 - 06902 Sophia Antipolis Cedex, France
{Erik.Pernod, Jean-Christophe.Souplet, Xavier.Pennec}@sophia.inria.fr
[2]Univ. Nice - Sophia Antipolis / CNRS - I3S Laboratory, BP 145 - 06903 Sophia Antipolis Cedex, France
{javier, lingrand}@i3s.unice.fr

Abstract

Automatic brain MRI segmentations methods are useful but computationally intensive tools in medical image computing. Deploying them on grid infrastructures can provide an efficient resource for data handling and computing power. In this study, an efficient implementation of a brain MRI segmentation method through a grid-interfaced workflow enactor is proposed. The deployment of the workflow enables simultaneous processing and validation. The importance of parallelism is shown with concurrent analysis of several MRI subjects. The results obtained from the grid have been compared to the results computed locally on only one computer. Thanks to the power of the grid, method's parameter influence on the resulting segmentations has also been assessed given the best compromise between algorithm speed and results accuracy. This deployment highlights also the grid issue of a bottleneck effect.

1 Introduction

The segmentation of lesions on brain MRI is required for diagnosis purpose in multiple Sclerosis (MS) [15]. Moreover, the lesion burden is also used in MS patients' follow-up and in MS clinical researches [5]. Different methods of lesions segmentation are available in the literature [18]. Most of them are based on complex algorithms and require numerous computations. Furthermore, medical images treatments need more and more computation power due to the increase of image size and resolution. Medical image databases also contain an increase amount of MRI subjects to analyze. The use of grids might help us to reduce computation time. For example, an evaluation framework for analyzing the accuracy of rigid registration algorithms has been made possible using a grid [9]; Action potential propagation on cardiac tissue simulations have also been performed on grid to accelerate multiple executions [1].

In this paper, we focus on the MS application aiming at segment MS lesions in brain MRI images. This work is part of the NeuroLOG project[1] [11] which aims at federating medical data, metadata and algorithms, and sharing computing resources on the grid.

This MS lesion segmentation algorithm has been developed by Dugas *et al.* [2, 3]. First, brain MRIs are normalized (spatially and in intensity) and skull-stripped. Then, a segmentation of the brain into the dif-

[1]NeuroLOG, http://neurolog.polytech.unice.fr

ferent healthy compartments classes (White Matter (WM), Gray Matter (GM), Cerebro-Spinal Fluid (CSF)) is realized using an Expectation Maximization algorithm. Resulting segmentations are used to segment lesion on the T2-FLAIR sequence. The expectation-maximization algorithm consists in iterating two steps: labelization of the image (Expectation step) and estimation of the Gaussian class parameters (Maximization step). In this last step, the class parameters are computed from the intensities of the different voxels. In order to improve the algorithm speed, only a part of the image voxel can be taken into consideration thanks to a ratio parameter. This parameter fix the percentage of voxel which is used. However, if the algorithm is applied on less voxels, the class parameters change and therefor the segmentation too.

To assess the influence of the ratio parameter, the deployment of the pipeline (until the brain segmentation), on the EGEE production grid using MOTEUR [8] as interface, will be presented. First, the brain segmentation algorithm will be acquainting as well as grid and MOTEUR tools. Then, the deployment of the pipeline will be described and validated. Finally, the grid will be used to assess the influence of the ratio parameter on the algorithm.

2 The Brain MRI segmentation pipeline: a preliminary step towards MS-lesions detection

The segmentation and the characterization of healthy tissues in multi-spectral MRI is the first step in order to separate them from lesions. This section describes the pipeline used for the brain segmentation.

The database of patient images is consistent: Each patient data set is composed by MRI sequences T1, T2 and Proton Density (PD) weights. The data is processed following the pipeline illustrated by Figure 1. We detail here the main sub-processes.

Registration

T2 and PD sequences are intrinsically co-registered but this is not the case of T1 which has moreover a higher resolution. To limit partial volume effect caused by the re-sampling, a rigid registration of T1 on T2 is performed using the algorithm described in [14]. Furthermore, the classification algorithm needs initial values of the probability of each voxel to belong to one of the healthy tissue compartment. This is given by the MNI atlas[2] but imply an affine registration of the atlas on the patient data.

Skull-stripping

To isolate brain healthy compartments, the skull-striping method described in [4] is applied.

Expectation Maximization

The first call to the EM-classification in our pipeline is for the intensity normalization. MRI images are often affected by bias [17]. A first classification (with the EM framework) of the brain into WM, GM and CSF classes is realized without bias compensation. From these segmentations, the parameters of the bias field are computed.

Later, the EM framework is used once again to classify brain MRI voxels from unbiased images. The EM algorithm gives the probability for each voxel to belong to each class. To obtain binary segmentations, each voxel is classified to the most probable class.

Unbias

Using parameters extracted from the first EM classification, intensities are corrected using the Expectation Maximization (EM) framework described in [16].

[2] http://www2.bic.mni.mcgill.ca

3 The EGEE grid and MOTEUR

A grid is a network of shared computing and storage resources connected in a grid topology [6]. In this paper, experiments were done on the EGEE production grid[3](Enabling Grid for E-sciencE), the largest multi-disciplinary grid infrastructure in the world, which connects more than 68000 CPUs to some 8000 users.

We express pipelines as workflows of services, described in the Scufl language (using the Taverna designer [13]). Each service corresponds to one of the sub-process described earlier and presents inputs and outputs that are connected together.

A Web-Service Description Language (WSDL) file describes a Web-Service by specifying two kinds of XML tags: Tags describing what has to be invoke and tags describing how to invoke it. In our framework, a generic web service description is used. Then, services are wrapped using a Generic Application Service Wrapper (GASW) [7].

Operators acting on the data flow itself define the iteration strategies over the input port of a service. In fact, when a service owns two inputs or more, an iteration strategy defines how to compose data from different inputs. Two different data composition operators have to be considered: the one-to-one (Dot operator) and the all-to-all (Cross operator) data composition operator [12].

The services are executed on the EGEE grid through the MOTEUR enactment engine [8], hiding to the user the complexity of individual services submissions and management.

The MOTEUR enactment engine allows three different kind of parallelism that are relevant for our application:

- Workflow parallelism which corresponds to the execution of two self-supporting services on two independent data (the different images can be unbiased in parallel).
- Data parallelism which corresponds to the competitor execution of independent data by a single service (the EM-classification using ratio parameter p_1 can be run in parallel of the EM-classification using ratio parameter p_2).
- Service parallelism which corresponds to the concurrent execution of two independent data items by two services linked by a precedence constraint (the first EM-classification using parameter p_1 can be run in parallel with the skull-stripping of data from next patient).

4 Gridification of the application

This section describes the creation of the MS brain MRI segmentation workflow step by step, from the used pipeline to the executable workflow. Then results validations and time performances are proposed. And finally, thanks to the possibility of large multiple executions given by the grid, a study of the influence of the ratio parameters (from the classification step 2) on the results has been done.

[3]EGEE, http://www.eu-egee.org

4.1 Creation of the workflow

For enabling the execution of services workflow, it is first necessarily to describe our pipeline as a workflow. Concretely, the pipeline has to be splitted into services correctly linked together. Then, iteration strategies have to be described between services in order to structure the workflow.

Splitting the pipeline The brain segmentation pipeline has been divided into different blocks, reflecting the description given in 2 and the already existing division of the pipeline in different legacy applications. For each block, inputs and outputs have clearly been defined, in order to define the corresponding service.

Web-service generation For each service, a Scufl file has been created including: an enumeration of inputs and outputs, as well as the name and the localization of the binary file, corresponding to this service. And if necessarily the name and the localization of the shell file script encapsuling the binary file.

In practice additional shell scripts are needed in some cases. In fact, binary files may have many outputs or may take fixed filenames as inputs. However, for MOTEUR, outputs from a service have to be listed and are automatically renamed. Consequently, shell scripts provide a good solution to overcome these problems. Furthermore, this is also helpful in case of different calls of the same service's definition. For example, three registrations are needed for the MS lesion segmentation algorithm 2 and they don't have the same list of arguments. But only one description of the service is used with a text file (listing the parameters) as an additional input. Then, these files are read in the shell script in order to correctly execute the software.

Services validation Before any transformation of the pipeline into a workflow, a first validation of the services has been made. They all have been independently executed and tested on the grid.

These tests reveal that even if the software are written in a generic language like C++, compiling it directly on a Computing Element (CE) of the EGEE grid without shared libraries is highly advised to avoid any kind of execution problems.

Workflow structure creation Then, Taverna has been used to structure the workflow: images and parameters text files have been defined as inputs, the four different healthy compartments classes as outputs and all services have been correctly linked together and to their corresponding Web-Service description. Figure 1 is a simplified scheme of the workflow.

Iteration strategies To allow consecutive execution of the workflow, each service input has been correctly composed with Dot and Cross product operators. Data concerning a patient, either workflow inputs or intermediate results, are composed with a dot product to avoid cross-road composition from images from one patient with images from another and are then composed with a cross product with others data. A tag is used in the input file to express the fact that tagged inputs are referring to the same patient. For example, in the case of the rigid registration of T1 on T2 sequence. Three different cases are possible with two patients (A and B):

- All inputs have the same tag. So they are composed by Dot products. In this case we will have in inputs: $\{T1_A, T1_B\}; \{T2_A, T2_B\}$ and $\{parameters_A, parameters_B\}$. And the results will be obtained from the compositions: $\{T1_A, T2_A, parameters_A\}$ and $\{T1_B, T2_B, parameters_B\}$

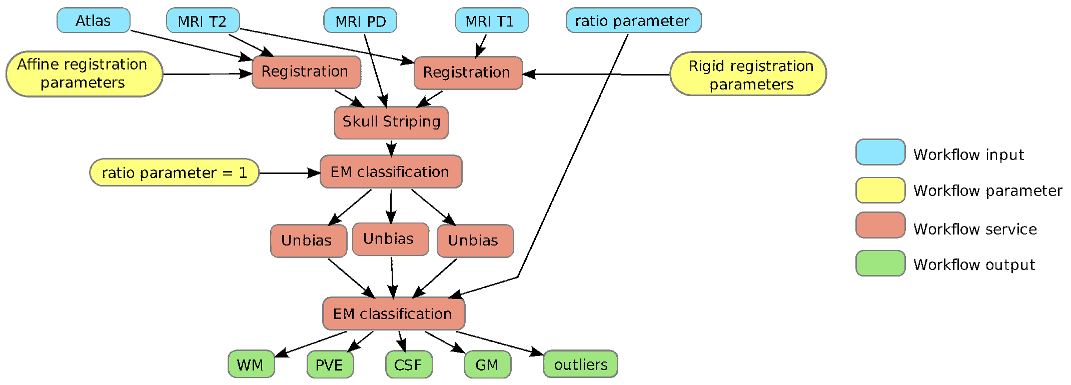


Figure 1: Simplified workflow of the brain segmentation process: a preprocessing step for multiple sclerosis detection.

- T1 and T2 only have the same tag. So they have to be composed by a Dot product and then are composed with a Cross product with the input parameters. In this case we will have in inputs: $\{T1_A, T1_B\}; \{T2_A, T2_B\}$ and $\{parameters\}$. And the results will be obtain from the compositions: $\{T1_A, T2_A, parameters\}$ and $\{T1_B, T2_B, parameters\}$

- All inputs are composed with Cross product. In this case we will have in inputs: $\{T1_A, T1_B\}$; $\{T2_A, T2_B\}$ and $\{parameters_A, parameters_B\}$. And the results will be obtain from the compositions:

  $\{T1_A, T2_A, parameters_A\}; \{T1_A, T2_A, parameters_B\}; \{T1_A, T2_B, parameters_A\}$;
  $\{T1_A, T2_B, parameters_B\}; \{T1_B, T2_A, parameters_A\}; \{T1_B, T2_A, parameters_B\}$;
  $\{T1_B, T2_B, parameters_A\}$ and $\{T1_B, T2_B, parameters_B\}$

4.2 Workflow execution and validation

In this section, we compare results obtained locally and from the grid. For this purpose, a ratio parameter equal to 1 has been used (all voxels from the image are taken for the maximization step *cf.* 4.3) and executions were done with images of 256×256×64 voxels for T2 and PD sequences and 256×256×152 for T1 sequence (see Figure 2). We verified that the results are absolutely identical.

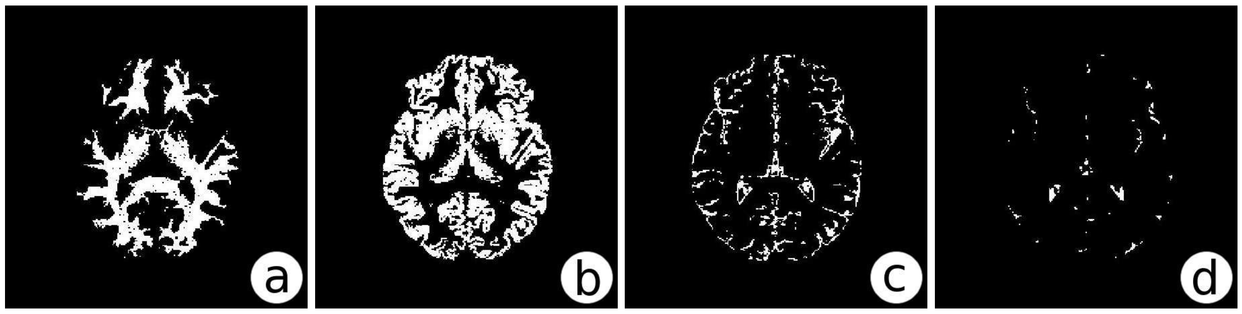


Figure 2: Output binary segmentations from the workflow on the EGEE grid : a) White matter, b) Grey matter, c) CSF, d) PVE.

Time performance Local executions have been done on a 2.0 GHz computer. We report the mean value over many "one-patient executions". As expected the local execution time evolves linearly in function of the number of input datasets (Figure 3). This is not the case for grid execution time. First, the execution time of the workflow can change depending on the resources availability and variations. To address this point, the workflow has been run several time on the same number of input datasets and the mean value has been computed. Secondly, the execution time is not constant (as it could be expected in case of complete parallelism). Indeed, the multiplication of inputs generate more transfers, so waiting time is more consequent. We can observe that for this application, using the grid appear to be efficient for more than 5 or 6 input datasets.

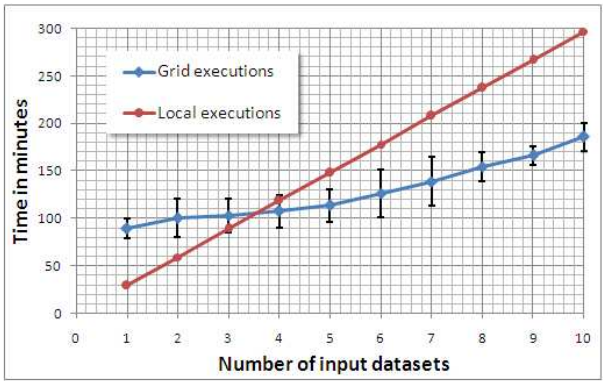


Figure 3: Mean execution time and its variations with respect of the number of input datasets. Comparison between local execution on a single computer and EGEE grid executions. Concerning EGEE grid executions, all points have been computed 3 times in order to compensate the workload variations.

Potential issues The EGEE grid is actually using the gLite middleware[4] which provides a framework for building grid applications. In this framework, the Resource Broker (RB) is among others in charge of accepting user jobs and then assigning them to the most appropriate CE. This choice is done by selecting CE which, first fulfill the requirements expressed by the user and then have the highest rank. On the EGEE grid, it appears that the rank is reflecting the response time of a CE. This choice can be discussed because the fastest responding CE is not necessarily the most powerful one, and above all, not necessarily directly available. Hence the workload management becomes sometimes a bottleneck for our application.

Moreover, the workload on the EGEE grid is highly variable thus leading to high and variable latencies and many faults, impacting the total execution time. After a delay, jobs have to be canceled and resubmitted. Optimizing job submission strategies is still on a research stage [10].

Of course, in between single computer and production grid, we could have envisaged the use of a cluster, improving the execution time for several (5 or 6) datasets without encountering production grid problems. However, this is not in our scope for different reasons. The first reason is we have in focus to be able to support large databases of patients; increasing the number of patients will conduct to cluster saturation. The second reason is that we are targeting final users that do not necessary have access to a cluster, even for small extend experiments; managing a grid access is thus a solution.

[4]Lightweight Middleware for Grid Computing, `http://glite.web.cern.ch/glite/`

4.3 Application

In the EM step the maximization consists in the estimation of the Gaussian parameters for each healthy tissue compartment class. These assessments are computed from the voxels intensities of the MRI. A ratio parameter define the fraction of voxel to be used (*e.g.* if the ratio is equal to 1 then the 100% of the voxel is considered). In this part, we will use the percentage of voxels considered. The relation between this percentage and the ratio is given by equation 1:

$$\text{percentage of voxel considered} \quad = \quad 100 * \frac{1}{\text{ratio parameter}} \tag{1}$$

In this section, we are targeting to assess the influence of ratio parameter on the pipeline results (Figure 2). In fact, by taking only a part of the image voxel, the speed of the algorithm could be improved but it could also affect the accuracy of the resulting segmentations. This is reason for studying the relationship between this parameter and the compromise between accuracy and speed is interesting for further works. To quantitatively evaluate this impact, WM segmentations have been generated for different ratio and have been compared to the segmentation of reference (*i.e.* ratio equal to 1) by computing the sensitivity and the specificity described in equation 2:

$$\begin{aligned} \text{sensibility} &= \frac{\text{true positives}}{\text{true positives}+\text{false negatives}} \\ \text{specificity} &= \frac{\text{true negatives}}{\text{true negatives}+\text{false positives}} \end{aligned} \tag{2}$$

Executions This experiment was made on images from two different patients affected by Relapsing/Remitting Multiple Sclerosis and one normal subject. The results of this experiment are similar. The graphic presented as illustration of this section was obtained from one MS patient.

It is important to underline that voxels are chosen randomly in the 3D image. Consequently, different results can be obtained for a same ratio parameter. To minimize the influence of this effect, many executions have been done and means values of the sensibility and the specificity have been computed. The Figure 4 displays these values in function of the percentage of voxel considered with the variations around mean values.

For this application, the power of the grid provides an efficient help to generate all the results (9 executions per ratio value). Indeed, the ratio parameter was written in an input parameter text file and has been assimilated as a relative to the patient (see Figure 1). Acting this way allows us to test all the different ratio parameters with each patient's MRI (case 1 of the iteration strategies in4.1).

Discussion Due to the skull-stripping step, the segmentation of the different healthy compartments is done on approximately 830.000 voxels. On Figure 4, we observe that the sensibility is decreasing while the percentage of voxels considered is decreasing. The specificity is more stable but those two quantities are more and more variable. Taking less than 1% of the voxels in our algorithm leads to results with too high variability: we cannot accept that different execution (with random voxel selections) lead to different results.

First, in this case, a WM segmentation with a specificity of 100% would mean that each voxel defined as belonging to (resp. not to) the white matter is really belonging to (resp. not to) the white matter in the segmentation of reference. But this doesn't mean that our segmentation results are accurate for low percentage ratio. Indeed, in our case, specificity and accuracy should not be confused because there are far more true negatives (voxels out of brain) than true positives (voxels really belonging to WM).

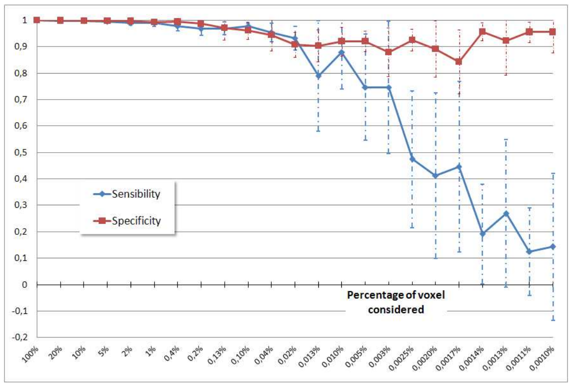


Figure 4: Mean sensibility and specificity of white matter segmentations in function of the percentage of voxel considered, and their variations. Each point corresponds to a mean of 9 executions (where voxels are randomly chosen).

Secondly, the drastic decrease of the sensibility means an increase of the number of false negative which corresponds to the voxel really belonging to the WM but not labelled as such. This reveals that after a certain threshold value of the ratio, there are not enough voxels any more in order to be able to define the Gaussian class parameter from the class estimation step of the EM.

Finally, these results reveal that using only 1% of the voxels of the image in the Expectation Maximization method would divide its execution time by 3 or 4 (compared to the execution with 100% of the voxels), without impacting the WM segmentation quality (Figure 5).

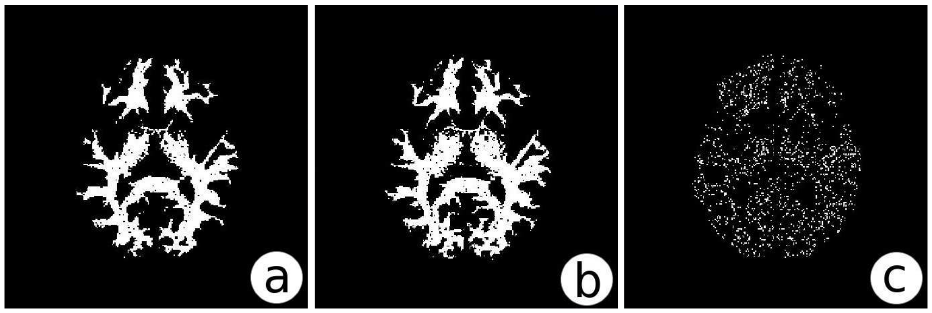


Figure 5: White matter binary segmentations from the workflow for different ratio percentage values : a) 100%, b) 0.2%, c) 0.002%.

5 Conclusion and future work

In this paper, a description of our method of brain segmentation into healthy compartments classes and its deployment on the EGEE grid has been presented. Experiments demonstrate that this application is well adapted to grid and provide a sizeable gain of time in multiple executions. Results of the workflow have been confronted to local results and have been successfully validated. Moreover, the power of the grid allows us to test the limit of our method of segmentation in order to improve the algorithm speed.

Our main finding is that in the expectation-maximization algorithm, taking only a part of the voxels doesn't affect severely the estimation of the Gaussian class parameter until a critical value. Thus, if needed, the brain healthy compartments classes could be generated faster while keeping a good accuracy. Indeed, tests have been repetitively done on a same patient with different value of the ratio of voxels and segmentation have been then compared.

The result of this experiment may be used for the following step which is the deployment of the segmentation of MS lesion, in the framework of the project NeuroLOG. This application is using the brain healthy compartments classes to segment lesion on the T2-FLAIR sequence. Future work will also improve the workflow execution speed on the grid, regrouping small services (to lower the number of resource requests) and testing different gLite parameters to increase the performance.

Acknowledgment

This work is funded by the French National Research Agency (ANR), NeuroLOG project, under contract number ANR-06-TLOG-024.

References

[1] J. M. Alonso, Jr. Ferrero, V. Hernández, G. Moltó, J. Saiz, and B. Trénor. A grid computing-based approach for the acceleration of simulations in cardiology. *Information Technology in Biomedicine, IEEE Transactions on*, 12:138–144, 2008. 1

[2] G. Dugas-Phocion, M. Ángel G. Ballester, G. Malandain, C. Lebrun, and N. Ayache. Improved EM-based tissue segmentation and partial volume effect quantification in multi-sequence brain MRI. In *Proc. of MICCAI'04*. Springer LNCS 3216, 2004. 1

[3] Guillaume Dugas-Phocion. *Segmentation d'IRM Cérébrales Multi-Séquences et Application à la Sclérose en Plaques*. PhD thesis, École des Mines de Paris, March 2006. 1

[4] Guillaume Dugas-Phocion, Miguel Ángel González Ballester, Christine Lebrun, Stéphane Chanalet, Caroline Bensa, Grégoire Malandain, and Nicholas Ayache. Hierarchical segmentation of multiple sclerosis lesions in multi-sequence MRI. In *International Symposium on Biomedical Imaging: From Nano to Macro (ISBI'04)*, Arlington, VA, USA, April 2004. IEEE. 2

[5] Massimo Filippi, Marco Rovaris, Matilde Inglese, Frederik Barkhof, Nicola De Stefano, Steve Smith, and Giancarlo Comi. Interferon beta-1a for brain tissue loss in patients at presentation with syndromes suggestive of multiple sclerosis: a randomised, double-blind, placebo-controlled trial. *Lancet*, 364(9444):1489–96, October 2004. 1

[6] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Jounral of Supercomputer Applications*, 15(3), 2001. 3

[7] Tristan Glatard, David Emsellem, and Johan Montagnat. Generic web service wrapper for efficient embedding of legacy codes in service-based workflows. In *Grid-Enabling Legacy Applications and Supporting End Users Workshop (GELA'06), Paris, France, June 19-23*, 2006. 3

[8] Tristan Glatard, Johan Montagnat, and Xavier Pennec. Efficient services composition for grid-enabled data-intensive applications. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing (HPDC'06), Paris, France, June 19*, 2006. 1, 3

[9] Tristan Glatard, Xavier Pennec, and Johan Montagnat. Performance evaluation of grid-enabled registration algorithms using bronze-standards. In *Proc. of the 9th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI'06), Part II*, number 4191 in LNCS, pages 152–160. Springer, 2-4 October 2006. PMID: 17354767. 1

[10] Diane Lingrand, Johan Montagnat, and Tristan Glatard. Estimating the execution context for refining submission strategies on production grids. In *Workshop ASSESS / Modern BIO (CCgrid'08)*, pages 753 – 758, Lyon, May 2008. IEEE. 4.2

[11] Johan Montagnat, Alban Gaignard, Diane Lingrand, Javier Rojas Balderrama, Philippe Collet, and Philippe Lahire. NeuroLOG: a community-driven middleware design. In *HealthGrid*, Chicago, June 2008. IOS Press. 1

[12] Johan Montagnat, Tristan Glatard, and Diane Lingrand. Data composition patterns in service-based workflows. In *Workshop on Workflows in Support of Large-Scale Science (WORKS'06)*, Paris, France, June 2006. 3

[13] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R. Pocock, Anil Wipat, and Peter Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004. 3

[14] S. Ourselin, A. Roche, S. Prima, and N. Ayache. Block matching: A general framework to improve robustness of rigid registration of medical images. In A.M. DiGioia and S. Delp, editors, *Third International Conference on Medical Robotics, Imaging And Computer Assisted Surgery (MICCAI 2000)*, volume 1935 of *Lectures Notes in Computer Science*, pages 557–566, Pittsburgh, Penn, USA, octobre 11-14 2000. Springer. 2

[15] C.H. Polman, S.C. Reingold, G. Edan, M. Filippi, H-P. Hartung, L. Kappos, F.D. Lublin, L.M. Metz, H.F. McFarland, P.W. O'connor, M. Sandberg-Wollheim, A.J. Thompson, B.G. Weinshenker, and J.S. Wolinsky. Diagnostic criteria for multiple sclerosis: 2005 revisions to the "Mc Donald Criteria". *Ann Neurol*, 58(6):840–6, 2005. 1

[16] S. Prima, N. Ayache, T. Barrick, and N. Roberts. Maximum likelihood estimation of the bias field in MR brain images: Investigating different modelings of the imaging process. In W.J. Niessen and M.A. Viergever, editors, *4th Int. Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI'01)*, volume LNCS 2208, pages 811–819, Utrecht, The Netherlands, Oct. 2001. 2

[17] J.G. Sled, A.P. Zijdenbos, and A.C. Evans. A nonparametric method for automatic correction of intensity nonuniformity in MRI data. *IEEE Trans Med Imaging*, 17(1):87–97, 1998. 2

[18] Jean-Christophe Souplet, Christine Lebrun, Stéphane Chanalet, Nicholas Ayache, and Grégoire Malandain. Revue des approches de segmentation des lésions de sclérose en plaques dans les séquences conventionnelles IRM. *Revue Neurologique*, 2008. 1

Large scale fMRI parameter study on a production grid[1]

Remi S. Soleman[1], Tristan Glatard[3,4], Dick J. Veltman[2], Aart J. Nederveen[1], and Sílvia D. Olabarriaga[3,4]

[1] Radiology, [2] Psychiatry and [3] Clinical Epidemiology, Biostatistics and Bioinformatics
Academic Medical Center, University of Amsterdam, NL
[4] Institute of Informatics, University of Amsterdam, NL

Abstract

Functional magnetic resonance imaging (fMRI) analysis is usually carried out with standard software packages (e.g., FSL and SPM) implementing the General Linear Model (GLM) computation. Yet, the validity of an analysis may still largely depend on the parameterization of those tools, which has, however, received little attention from researchers. In this paper, we study the influence of three of those parameters, namely (i) the size of the spatial smoothing kernel, (ii) the hemodynamic response function delay and (iii) the degrees of freedom of the fMRI-to-anatomical scan registration. In addition, two different values of acquisition parameters (echo times) are compared. The study is performed on a data set of 11 subjects, sweeping a significant range of parameters. It involves almost one CPU year and produces 1.4 Terabytes of data. Thanks to a grid deployment of the FSL FEAT application, this compute and data intensive problem can be handled and the execution time is reduced to less than a week. Results suggest optimal parameter values for detecting amygdala activation, as well as the robustness of results obtained for the difference between two studied echo times.

1 Introduction

Functional magnetic resonance imaging (fMRI [12]) is a noninvasive method for detecting brain activation that is now applied extensively in neuroscience, neurosurgical planning and drug research. fMRI detects changes in oxyhaemoglobin/deoxyhaemoglobin ratio resulting from increased local perfusion in the brain following a rise in neural activity, the so-called blood oxygenation level dependent (BOLD) contrast. Functional MR data can be acquired rapidly and spatial resolution is high, so that large data sets are generated. fMRI data is usually analysed with software packages such as fMRIB Software Library (FSL) [13] and Statistical Parametric Mapping software (SPM) [8]. Although the GUI of these packages conceals much of the complexity of the image analysis process, the choice of parameters still plays an important role in fMRI analysis. Most researchers perform the analysis using standard (default) parameter settings without questioning the role of these parameters. Comparing results for different parameter setting is in most cases impractical due to the huge amount of computing resources required for analysis and data storage.

[1]This work was carried out in the context of the VL-e project, which is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W) and is part of the ICT innovation program of the Ministry of Economic Affairs (EZ).

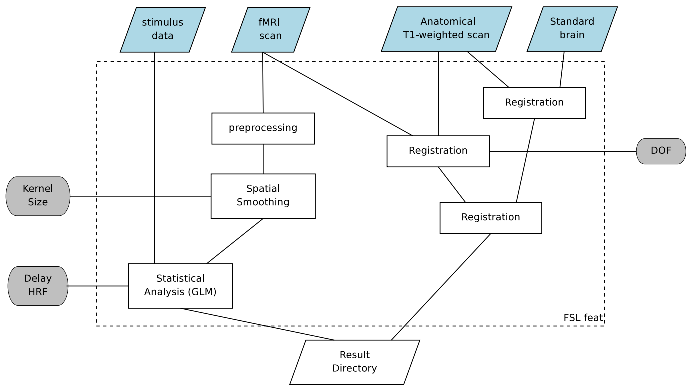


Figure 1: Simplified representation of FSL FEAT (fMRI Expert Analysis Tool) emphasising steps that are most relevant in this study. White boxes are processing steps, blue boxes are input files, and grey boxes are parameters investigated in this study.

In this paper we present a practical example of a parameter study in fMRI in which we varied three different parameters in a standard data preprocessing procedure and General Linear Model (GLM) analysis. We used for this example an emotional-provoking task which is known to activate the amygdala: a brain area mainly involved in proccessing of emotional reactions and memory. We also investigate the effect of one image acquisition parameter, namely echo time. The analysis is performed on a grid infrastructure, enabling this compute and data intensive problem to be tackled within a reasonable amount of time. This example illustrates the usefulness of such methodological experiments that employ massive computing resources to investigate the optimal parameters settings and the robustness of results obtained with fMRI.

The paper is organised as follows. Section 2 presents details of the fMRI problem and the designed parameter sweep experiment. The implementation of this experiment on a grid infrastructure is presented in Section 3. fMRI and grid performance results are presented and discussed in Sections 4.1 and 4.2. Section 5 presents initial conclusions of this study.

2 fMRI Parameter Study

In this study the data was analysed with FSL, version 3.3 [13], using its software tool fMRI Expert Analysis Tool, version 5.63 (FEAT). Each fMRI data set is first individually submitted to first-level analysis, which calculates brain activation maps as a result of several image analysis steps. The most relevant ones for this study are illustrated in figure 1 and briefly presented below. After pre-processing (including motion correction, brain extraction and slice-timing), the fMRI scans are spatially smoothed using a kernel of size S. Then, the GLM analysis is performed taking into account the HRF delay H. In GLM, a model is created to fit the fMRI scan data with respect to the timing of the stimulation paradigm employed during the scanning session. It is assumed that a good fit between the model and the data means that changes in BOLD-signal are causally related to the stimulation paradigm [12]. The output of an fMRI analysis is a brain activation map containing standardised activation probabilities (Z-scores). Normalisation to the standard brain is performed in two steps, the first of which being the alignment of the fMRI data to the anatomical scan taking into account a number of degrees of freedom D. The parameters of interest are D, S and H, which were investigated with a parameter sweep experiment.

2.1 Parameters of Interest

The first part of the parameter sweep concerns the normalisation to a standard brain. To compare individual scans at a group level it is necessary to align the individual scans to a predefined template brain. In this study, it is the standard MNI152 brain distributed with FSL. As shown in figure 1, the normalisation to the standard brain is performed through 3 different registrations. First, the high-resolution T1-weighted scan is mapped to the standard brain and to the fMRI scan using two independent registration procedures. Then, those two results are combined to produce the fMRI-to-standard-brain mapping. The **degrees of freedom (*D*)** investigated in this study only concern the fMRI-to-T1 registration. The fMRI-to-standard-brain registration is always performed using 12 degrees of freedom. Depending on the amount of correction needed it is possible to restrict manually the transformation type by adjusting D to a lower level (in FSL one can choose between 12, 9, 7, 6 or 3) to prevent failures (e.g. complete flip of an image). A wrong choice of restriction will result in a poor registration and different brain areas will be compared with each other.

The second part of the parameter sweep concerns spatial smoothing. Reducing the spatial resolution of fMRI data increases the signal to noise ratio (SNR), which improves the sensitivity. On the other hand, when using large **smoothing kernels (*S*)** the spatial resolution will be lost without gaining any profit with regard to sensitivity. In many cases a smoothing kernel $S = 6$ mm (full width at half maximum) is used for a typical voxelsize of approximately 2-3 mm.

The third part of the parameter sweep concerns the **delay of hemodynamic response (*H*)**. Functional MRI provides an indirect measure of brain activity, since fMRI detects changes in blood oxygenation level resulting from increased local perfusion following a rise in neural activity. Therefore, fMRI analysis need to correct for this delayed hemodynamic response. This is performed by convolving the modeled activity pattern in the brain with a hemodynamic response function (HRF). In general a hemodynamic delay $H = 6$ s is used, although there is evidence that this value may vary within and between subjects [15, 1].

Apart from the effects of parameter manipulation, we are also interested in comparing MRI sequences. An important parameter in an MRI sequence is the **echo time (*T*)**, indicating the time window between the transmission of a radiofrequency pulse and the signal acquisition in fMRI. In general, the usage of a sequence with a shorter echo time will result in higher signal and smaller susceptibility artifacts in the images, whereas the contrast between high and low brain activity states will tend to decrease [6]. However, it is difficult to predict from theoretical considerations which acquisition scheme should be used to obtain optimal results. In addition we suspect that the difference between these two protocols might also depend on the parameter values used for fMRI analysis.

2.2 Experiment Design

Data Acquisition. To assess both research questions we used data of healthy volunteers acquired on a Philips 3.0 Tesla Intera scanner during an event-related task paradigm (viewing of emotional pictures, International Affective Picture System (IAPS) [9]). The task was presented twice during a session with different echo time: 28 ms and 35 ms. The order of presentation was counterbalanced between subjects. Whole brain imaging was performed by scanning axial slices with an in-plane resolution of 1.9 mm and a slice thickness of 3.0 mm. For $T = 28$ ms and $T = 35$ ms we used a repetition time (time between successive volumes) of 2.7 s and 3.1 s respectively. Except for the echo times and the repetition times, all scanning parameters were kept identical. In total 22 fMRI scans $F_{i,T}$ were acquired for 11 subjects $P_i, i \in [1, 11]$ with echo times $T \in \{28, 35\}$.

	CPU time	Input size (MB)	Results size (MB)
Indiv. analysis	49 min	104.5	150
Group analysis	8.6 min	1650	30.5
Group difference	22.8 min	3300	55.5

Table 1: Characteristics of the individual and group analyses. CPU times have been measured on a Dual-Core AMD Opteron 2613.427 MHz with 3.5 GB of RAM. Groups summarise results of 11 individuals.

Data Analysis. Each scan $F_{i,T}$ was analysed individually using FSL FEAT first-level analysis with varying parameters: HRF delay $H \in \{2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5\}$, smoothing kernel size $S \in \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$, and degrees of freedom for registration $D \in \{3, 6, 7, 9, 12\}$. Results of this phase consist of activation maps $\alpha_{i,T}(H,S,D)$ obtained with each combination of parameter values (H,S,D).

After all the individual activation maps were computed, an average activation map was calculated for groups of results using FSL FEAT high-level analysis. Results obtained for all P_i with identical (T,H,S,D) are averaged, generating group activation maps denoted as $\gamma_T(H,S,D)$. Finally, the results obtained with two echo times were compared using FSL FEAT group difference analysis in a similar way. The generated difference maps are denoted $\Delta(H,S,D)$.

Results Evaluation. The comparison of results obtained with different parameters was performed based on a region of interest with robust activation for the adopted stimulus. The IAPS is a picture set containing emotion-provoking pictures, both positive (e.g. erotic scenes) and negative (e.g., mutilations, attack scenes), which has often been used in emotion research. Previous studies have shown that viewing IAPS pictures vs. baseline activates regions within the brain that are involved in emotional processing such as the amygdalae [3, 10]. The amygdalae are almond shaped groups of nuclei within the left and right medial temporal lobe which are one of the key elements of emotion processing. Therefore, the mean Z-scores in both of the amygdalae were used as reference to compare activation maps obtained with different parameter settings. The amygdalae were located with a predefined anatomical atlas (AAL, [14, 7]) on the MNI152 template and the created mask was applied to the group activation maps $\gamma_{28}(H,S,D)$, $\gamma_{35}(H,S,D)$ and $\Delta(H,S,D)$, to extract the Z-scores within this region of interest and calculated their mean denoted $\mu_{28}(H,S,D)$, $\mu_{35}(H,S,D)$ and $\mu_D(H,S,D)$.

3 Grid implementation

The designed experiment involves large computation effort. The individual analysis is the most compute-intensive one due to pre-processing, registration and linear model computation. The group analysis is the most data intensive one, since it needs to access all the individual analysis results for averaging (See summary in table 1). Parameter sweeps are costly: in particular, when the parameter space has several dimensions (three in this study), widening the range of one of them swiftly increases the size of the computing problem. As a side effect, storing the produced data is often not manageable without a specific infrastructure. The total estimated resources for this study add up to almost one CPU-year and 1.4 Terabytes of data. Obviously, this experiment could not have been performed without a high performance infrastructure.

Production grids have been designed to support such computing and data needs, but they are still seldom used by medical imaging researchers for such studies. Roadblocks need to be identified and carefully targeted by

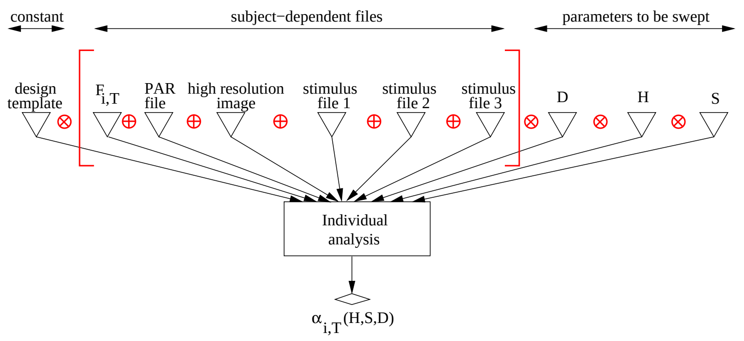


Figure 2: Individual analysis workflow described in Scufl. Iteration strategies are drawn in red ($\otimes$ is the cross-product and $\oplus$ is the dot product). Scufl iteration strategies allows for easily distinguishing sweeping parameters from data-related inputs.

the grid community with new developments. To do that, the Dutch Virtual Laboratory for e-Science (VL-e[1]) set up a tight interaction between domain scientists and grid developers and grid service providers. The project gives access to a Proof-of-Concept grid infrastructure that is part of EGEE. It also offers a fertile ground for exchange of expertise to develop and deploy grid-enabled applications in answer to real demands of domain scientists as it is the case in this fMRI study.

3.1 Grid Application Design

As part of the VL-e software distribution, the FSL package is pre-installed in all computing nodes, therefore scripts can be submitted directly as grid jobs to perform FSL FEAT or other image analysis tasks. The gLite command line utilities could have been used to submit and monitor jobs on the grid, however this approach was found to be too unfriendly. Instead, we adopted a workflow approach by wrapping image analysis tasks as web-services, composing workflows to describe the parameter sweep experiment, and enacting the workflows on the production grid. More details are presented below.

The Scufl workflow language from the Taverna workbench [11] was used to describe workflows. Even if this use-case does not exhibit strong workflow requirements (like complex data dependencies between components), using such a language is useful to describe the parameter sweep in an elegant way. In short, Scufl supports lists to be indicated as inputs, and the workflow is iterated for each list element. Operators specify how list elements should be combined as workflow inputs. The “cross product” $A \otimes B$ indicates that the workflow is iterated for all combinations of elements $a_i \in A$ and $b_j \in B$, and the “dot product” $A \oplus B$ indicates that the workflow is iterated for pairs (a_i, b_i). Using such a generic workflow framework instead of ad-hoc scripts is a step towards an autonomous usage of the grid by end-users, which is an important aim of our research. Two workflow descriptions were used to implement the parameter sweep experiment, one for the individual analyses and one for the group analyses.

The **individual analysis workflow** takes 10 inputs and produces a single output (see figure 2). The performed task consists of FSL FEAT first-level analysis and some data logistics (download/upload data). Six of the inputs correspond to the fMRI scan $F_{i,T}$ (100 MB) and other subject-related data: MRI acquisition parameters (1.5 MB), the high resolution anatomical (T1-weighted) scan (1.5 MB) and 3 files with stimulus timing information (500 bytes each). Three other inputs correspond to the parameters (H, S, D) to

[1]`http://www.vl-e.nl`

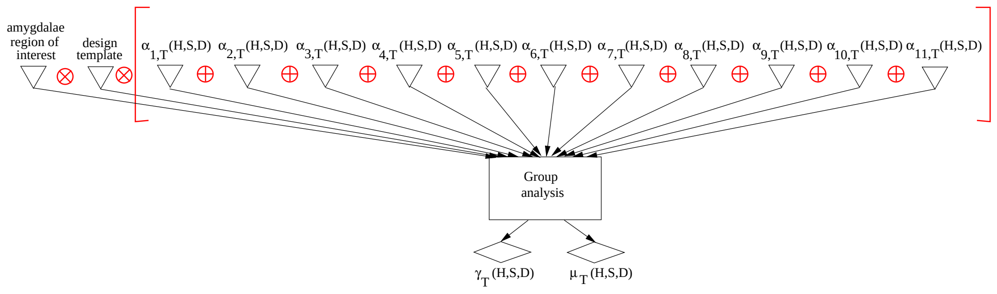


Figure 3: Group analysis workflow described in Scufl.

sweep upon. The last input (design file) contains all the parameters for FSL FEAT that remain constant in all analyses. Each individual analysis produces a single directory with all the output files (150 MB when compressed), one of them being the brain activation map $\alpha_{i,T}(H,S,D)$ used in group analyses. In this experiment, the inputs assume lists of values associated with different scans, subjects, and parameters. The iteration strategies of Scufl allow to express the sweep on individual analysis tasks in a simple way: the 6 first data-related inputs are combined with the dot product operator, and the other parameters are swept with the cross product.

The **group analysis workflow** used to calculate average activation has 13 inputs and two outputs – see figure 3. Besides running the FSL FEAT high-level analysis, this workflow also calculates the mean value in the amygdalae using with the FSL AVWMATHS utility. Eleven inputs indicate directories containing pre-computed results of individual analyses for 11 fMRI scans $F_{i,T}$. These inputs are provided by lists in which the elements correspond to directories with results computed with the same parameter settings (H,S,D). The dot product is used to guarantee that the activation maps averaged during the workflow iteration are $\alpha_{i,T}(H,S,D), i \in [1,11]$. The remaining inputs are constant in this experiment, namely the design file with parameters for FSL FEAT, and the mask indicating the voxels in the amygdalae. The group analysis produces 2 outputs: a directory with results (30 MB when compressed), including the group activation map $\gamma_T(H,S,D)$, and a file containing the mean Z-score $\mu_T(H,S,D)$ in the amygdalae region.

Similarly to the above, a workflow with 23 inputs and 2 outputs is used for **group difference analysis** to compare results obtained with different echo times T. Twenty two inputs indicate directories with pre-computed individual analyses corresponding to two groups fMRI data $F_{i,28}, F_{i,35}, i \in [1..11]$ acquired with different echo times. The outputs include the difference activation maps for these two groups $(\Delta(H,S,D))$ and the mean values in the amygdalae $(\mu_D(H,S,D))$.

3.2 Application Deployment

The grid infrastructure of the Dutch VL-e project was used to run the experiment. At the time the experiment was performed the `vlemed` Virtual Organisation (VO) had access to 687 CPUs spread over 4 sites through 8 batch queues federated by the gLite middleware. As a production grid, it is shared among several VOs, therefore there is no way to control the number of available CPUs at a given instant. Four Storage Elements (SE) accessible through an SRM interface are also available and files can be organised in a single Logical File Catalog (LFC) independently from their physical location. In addition, a gridFTP and a Storage Resource Broker servers are also available.

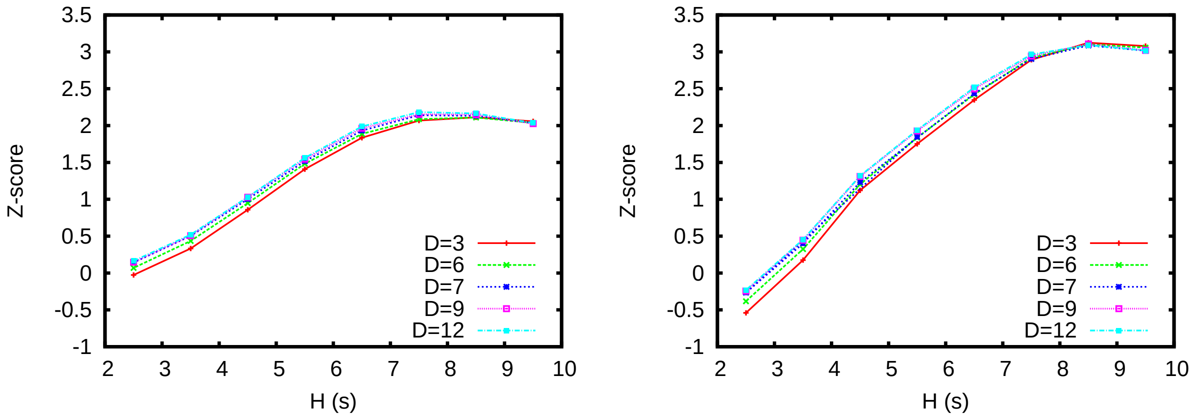


Figure 4: Influence of degrees of freedom D in the mean activation in the amygdalae as a function of delay for $T = 28$ ms using smoothing kernels $S = 5$ mm (left) and $S = 11$ mm (right).

Porting the image analysis application to the grid was easily done using the Generic Application Service Wrapper (GASW) [4], which integrates executables in Scufl workflows via a command-line descriptor and the Web Service Description Language (WDSL). The FEAT script was wrapped as shipped by the Fhttp://www.thefreedictionary.com/timeSL distribution. Workflows are executed on this infrastructure using the MOTEUR workflow engine [4] and adopting the software architecture detailed in [5]. The various storage resources are accessed via the homogeneous Virtual File System interface provided by the VL-e software Toolkit[2].

4 Results and Discussion

4.1 fMRI results

Figure 4 illustrates the mean activation within the amygdalae using different degrees of freedom D for the fMRI-to-anatomical scan registration. The plot shows that no significant activation differences were observed, a pattern that is also observed for other smoothing kernel sizes. In particular, it should be noted that a registration with $D = 3$ (translation only) produced similar results as obtained with $D = 9$ and $D = 12$. Given that both scans belong to the same subject and have been acquired during a single scanning session, only minor differences are expected, which could explain why only translations seem sufficient to capture them. The main contribution to the normalisation process is likely to be the anatomical-to-standard-brain registration, the second step in the registration process, which was fixed to 12 degrees of freedom in this study. A further analysis of the complete registration process will be part of our future work.

Figure 5-left shows the mean activation within the amygdalae obtained with different spatial smoothing kernel sizes and HRF delays. Note that a maximum for the Z-score is obtained for a delay of $H = 8.5$ s, which differs from the value adopted by default in FSL feat ($H = 6$ s). The figure also shows an activation increase when using larger spatial smoothing kernels. A visual inspection of the results (figure 6) suggests that with $S = 12$ mm spatial resolution is still acceptable and that even larger smoothing kernels might further increase sensitivity.

[2]`http://www.science.uva.nl/~ptdeboer/vlet`

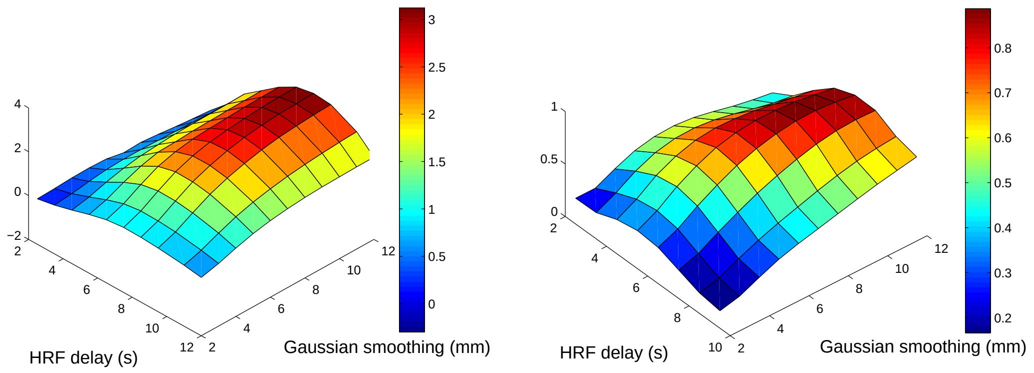


Figure 5: Results obtained for group analyses for varying smoothing kernel sizes S and HRF delays H. Left: Mean activation (Z-scores) in the amygdalae for $T = 28$ ms, $D = 12$. Right: Mean differences between amygdalae activation with $T = 28$ ms and $T = 35$ ms.

Our second research question concerns the effects of manipulating echo time (T). Pairwise T-tests were used to compare results obtained with both echo times. No significant differences in activation between the scan sequences could be detected, as shown in figure 5-right: none of the parameter combinations cross the threshold of significance. We used a Z-value of 2.33 (which equals a probability of 0.01) as threshold to distinguish noise from an actual signal. ([16]).

4.2 Grid performance

Table 2 presents a summary of performance results of the individual and the group analyses grid application as measured during the execution of this parameter study on the VL-e grid. A total of 9680 individual analyses, 880 group analyses and 440 group differences was computed. CPU time was benchmarked on a node of the infrastructure with representative performance.

Individual analyses were submitted in 4 batches of about 2500 jobs each to avoid infrastructure flooding. The global speed-up of 115 times enabled us to compute the individual analyses in less than 3 days. Note that this is a compute-intensive experiment, therefore the speed-up value provides an indication of the number of CPUs that were used concurrently. Note also the small job failure rate of 0.5% measured for this experiment.

Although pleasantly parallel, group analyses faced quite poor speed-ups (global 2.9). One of the reasons is they are dominated by data transfers, since all individual analyses are downloaded to the worker node before executing FSL FEAT. Even if each analysis only transfers about 1 GB, concurrently initiating hundreds of

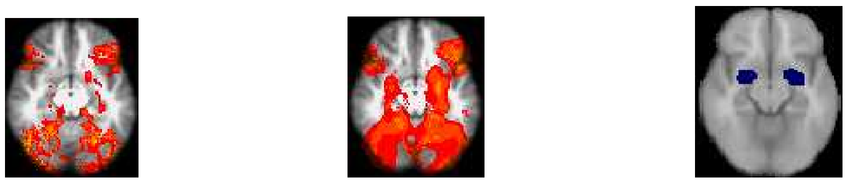


Figure 6: Impact of the smoothing kernel size for $H = 2.5$ s and $D = 12$. Left and Centre: slice of the brain activation map respectively for $S = 5$ mm and $S = 12$ mm smoothing. Right: slice of the amygdalae region of interest (in blue) overlaid on the template brain.

	$\#P_i$	#T	#S	#D	#H	# Analyses	CPU (days)	Data (TB)	Elapsed (hours)	Speed -up	# Submit Jobs	Failure (%)
Individual Analyses												
batch 1	11	1	5	5	8	2200	74.9	0.31	14.9	120.5	2200	0.00
batch 2	11	1	6	5	8	2640	89.8	0.38	11.6	186.6	2642	0.08
batch 3	11	1	6	5	8	2640	89.8	0.38	32	67.38	2687	1.75
batch 4	11	1	5	5	8	2200	74.9	0.31	10.2	176.8	2203	0.14
total	11	2	11	5	8	9680	329.4	1.38	68.7	115	9732	0.53
Group Analyses								(MB)				
batch 1		1	6	5	8	240	1.4	7.1	8.0	4.3	401	40.15
batch 2		1	6	5	8	240	1.4	7.1	9.5	3.6	240	0.00
batch 3		1	5	5	8	200	1.2	6	14.9	1.9	200	0.00
batch 4		1	5	5	8	200	1.2	6	11.3	2.5	600	66.67
total		2	11	5	8	880	5.2	26.2	43.7	2.9	1441	38.93
Group Difference Analyses								(MB)				
batch 1			11	5	8	440	7	23.8	44.3	3.8	2650	83.40

Table 2: Performance results for individual, group and difference analyses for each job batch: number of subjects, echo times, smoothing sizes S, degrees-of-freedom D, HRF delays H, and successful analyses; total CPU time, amount of produced data, total elapsed time from first submitted to last completed job, speed-up of the experiment (CPU/elapsed), and job failure rate ((#submitted - #successful) / #submitted).

such data transfers rapidly reduces the data servers throughput. Moreover, one of the Storage Elements limited the number of concurrent connections, causing jobs to fail and partially explaining the high 38.93% global failure ratio. This problem was even larger in the case of group difference analyses, where each of the 440 jobs concurrently attempted to download more than 3 GB. The global failure rate here was 83%. In spite of these problems, the experiment could be carried out in a reasonable time in the end, which could not have been possible without adopting a grid implementation. All in all, those failure ratios look good compared to other medical experiments conducted on EGEE (see *e.g.* [2]). Two main reasons can explain such a difference. First, our experiment is set up in a quite controlled environment, with pre-installed software and strong support team on every grid site. Second, it has been carefully tuned in order to avoid some of the problems encountered in [2]: for instance, the total workload has been split into batches of reasonable size to avoid proxy expirations.

5 Conclusion

In this paper, the benefit of performing parameter sweeps for methodological fMRI studies has been established by assessing amygdala activation for different analysis parameter choices. Detecting activation in this brain area is in general cumbersome, emphasising the need of a good choice of parameters to obtain reliable results. Our initial results indicate that the optimal values deviate from the default parameters that are typically used in this type of analysis in FSL FEAT. In the future we plan to perform this analysis for different fMRI paradigms and different brain areas and expect to find different optimal parameter values. Additionally, we could study the robustness of the difference between two different MRI scan protocols over different parameter sets for analysis. Results indicate that the observed difference between the considered echo times is not significant and does not depend on the parameters used for analysis.

Although the grid implementation enabled the experiment to be completed within a reasonable time, the set-up is still far from ideal. Data transfers, rather than computation, seem to be the limiting factor now and data distribution or even replication among several Storage Elements might be a way to improve their reliability. Then, a proper strategy would have to be determined in order to avoid blind replication of terabytes of data. Moreover, failure management is still a challenging issue in grid applications deployed in production grids, with the consequence that much (expert) manual intervention is still needed to complete the experiment. This not only causes a significant overhead in the management of this experiment, but also makes the dream of having end-users performing medical image analysis on grids seem further away than we initially hoped for. We continue to work on improvements in the grid application to support the many experiments to come motivated by the promising results obtained in the study reported here.

References

[1] G. Aguirre et al. The variability of human bold hemodynamic responses. *NeuroImage*, 8(4):360–369, 1998.

[2] G. Aparicio et al. A Highly Optimized Grid Deployment: the Metagenomic Analysis Example. In *Global Healthgrid: e-Science Meets Biomedical INformatics (Healthgrid'08)*, pages 105 –115, Chicago, USA, May 2008. Healthgrid, IOS Press.

[3] H. Breiter et al. Response and Habituation of the Human Amygdala during Visual Processing of Facial Expression. *Neuron*, 17(5):875–887, 1996.

[4] T. Glatard et al. A Service-Oriented Architecture enabling dynamic services grouping for optimizing distributed workflows execution. *Future Generation Computer Systems*, 24(7):720–730, 2008.

[5] T. Glatard et al. Workflow integration in VL-e medical. In *21st IEEE International Symposium on Computer-Based Medical Systems (CBMS'08)*, pages 144–146, Jyväskylä, Finland, June 2008.

[6] M. Gorno-Tempini et al. Echo Time Dependence of BOLD Contrast and Susceptibility Artifacts. *NeuroImage*, 15(1):136–142, Jan. 2002.

[7] C. Holmes et al. Enhancement of MR images using registration for signal averaging. *Journal of Computer Assisted Tomography*, 22(2):324–333, Apr. 1998.

[8] F. K.J. Statistical parametric mapping and other analysis of functional imaging data. In *Brain Mapping: The Methods*, pages 363–385. Academic Press, 1996.

[9] P. Lang et al. International affective picture system (IAPS): Technical manual and affective ratings. Technical report, University of Florida, Center for Research in Psychophysiology, Gainesville, 1999.

[10] I. Liberzon et al. Extended amygdala and emotional salience: a PET activation study of positive and negative affect. *Neuropsychopharmacology*, 28(4):726–33, Apr. 2003.

[11] T. Oinn et al. A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics journal*, 17(20):3045–3054, 2004.

[12] S. Smith. Overview of fMRI analysis. *The British Journal of Radiology*, 77:S167–S175, 2004.

[13] S. Smith et al. Advances in functional and structural MR image analysis and implementation as FSL. *NeuroImage*, 23(1):S208–S219, 2004.

[14] N. Tzourio-Mazoyer et al. Automated anatomical labelling of activations in spm using a macroscopic anatomical parcellation of the MNI MRI single subject brain. *NeuroImage*, 15(1):273–289, Feb. 2002.

[15] M. Woolrich et al. The variability of human bold hemodynamic responses. *NeuroImage*, 21:1748–1761, 2004.

[16] K. Worsley et al. A three-dimensional statistical analysis for CBF activation studies in human brain. *Journal of Cerebral Blood Flow and Metabolism*, 12:900–918, 1992.

Validation of the Small Animal Biospace Gamma Imager Model Using GATE Monte Carlo Simulations on the Grid

Joe Aoun[1, 2], Vincent Breton[2], Laurent Desbat[1], Bruno Bzeznik[3], Mehdi Leabad[4] and Julien Dimastromatteo[5]

[1]Grenoble Joseph Fourier University, TIMC-IMAG
[2]Clermont Ferrand Blaise Pascal University, LPC
[3]Grenoble Joseph Fourier University, CIMENT
[4]Biospace LAB, Paris
[5]Grenoble Joseph Fourier University, INSERM U877

Abstract

Monte Carlo simulations are nowadays widely used in the field of nuclear medicine. They are valuable for accurately reproducing experimental data, but at the expense of a long computing time. An efficient solution for shorter elapsed time was recently proposed: grid computing. The aim of this work is to validate a Monte Carlo simulation of the Biospace small animal γ Imager and to confirm the usefulness of grid computing for such a study. Simulated data obtained by the validated model of the gamma camera will enable to investigate new algorithms such as scatter and attenuation correction, and reconstruction methods. Good matches between measured and simulated data were achieved and a crunching factor as high as 70 was achieved on a campus grid.

Contents

1 Introduction 75

2 Materials and Methods 76

3 Results and discussions 80

4 Conclusions and perspectives 84

1 Introduction

Preclinical small animal researches have become a major focus in nuclear medicine [1]. New therapeutic and diagnostic studies are first investigated and validated on mice or rats before their application to patients. In consequence, we can find many dedicated small field of view scanners for SPECT (Single Photon Emission Tomography) [2, 3] and PET (Positron Emission Tomography) [4] which have been designed in the last decade for these purposes. SPECT images have a very poor quality because generally used models do not incorporate all physical interactions such as scattering (30% of photons in SPECT images are scattered). Monte Carlo Simulations (MCS) have greatly contributed to these developments thanks to their accuracy and utility in the field of nuclear medicine. The use of the MCS has limitations in computing time. Different strategies have been suggested to reduce the computing time such as the acceleration techniques [5]. Another solution to speed up MCS is to combine Monte Carlo and non-Monte Carlo modelling [5]. A third option that has recently been explored is the deployment of computing grids, also known as the parallelization of the MCS [6]. The parallelization consists in sub-dividing a long simulation into short ones. Each MCS uses a Random Number Stream (RNS) to produce the physical interactions in question. The distribution of MCS on multiple computing resources requires that the generated streams are independent. In this work, we report the validation of the Biospace dedicated to small field of view SPECT scanner using the GATE MC simulation toolkit on the CIMENT Grid, a grid deployed on the university campuses in Grenoble.

The paper is organized as follows:

Chapter 2 presents the materials and methods used in this study: we introduce CiGri (CIMENT Grid), the GATE MCS toolkit, the gamma camera we studied and the methods we used to deploy the simulations on the grid and to validate the simulated results compared to experimental measurements. Chapter 3 proposes an analysis of the results of simulation as well as an overall view of the grid performances. Chapter 4 presents a conclusion and perspectives.

2 Materials and Methods

2.1 CiGri grid

CiGri (CIMENT Grid) is a lightweight grid which exploits the unused resources of the CIMENT clusters on the university campuses in Grenoble (France) [7]. It manages by using the OAR resource management system [8], the execution of a large set of parametric tasks (typically more than 100K) by submitting individual jobs to each cluster batch scheduler. One of the rules is that CiGri jobs shall not disturb the local users. CiGri jobs are treated as a zero priority jobs. Thus, a CiGri job is executed only if there is an idle cluster resource. Furthermore, if a local user requests a resource on which a CiGri job is running, then the job will be killed and automatically resubmitted later probably on another cluster. This approach is based on the concept of “best-effort” tasks. CiGri software is composed of several independent modules interacting with an SQL database. Each module is in charge of a specific task: scheduling jobs, submitting jobs, cluster synchronizing, monitoring jobs, collecting results, logging errors and killing jobs. The CiGri infrastructure is accessible through a User Interface (UI). The accessibility was simplified by adopting the same services and configurations on all interconnected clusters, that’s why CiGri is called a lightweight grid. In order to launch a job, a JDL file (Job Description Language) and a multi-parameter file are requested. Output files are first collected automatically by CiGri from the different clusters, then gathered in a compressed file and finally put in an output directory on the UI, so that the user can easily retrieve them. Users have the possibility to monitor and to control their set of jobs through a web interface. CiGri middleware is a free software under GPL licence (http://cigri.imag.fr/).

2.2 Monte Carlo simulation toolkit

GATE (Geant4 Application for Tomographic Emission) [9] is a new Monte Carlo simulation toolkit based on the general-purpose simulation package Geant4. It was developed in order to model treatment and diagnosis examinations such as radiotherapy, SPECT and PET in the field of nuclear medicine. GATE is an open source code which uses approximately 200 C++ classes from Geant4. User does not have to program in C++ thanks to an extended version of Geant4 scripting version. Thus, the user can easily build a simulation (macro) by using a script language. Many researchers have been using GATE for its flexibility and its accurate modelling of different detector designs and very complex geometries. The major drawback of GATE is the computation time especially when simulating realistic configurations such as voxelized emission and attenuation maps. Several solutions were proposed to accelerate some of the GATE simulations: (*i*) setting thresholds for the production of secondary electrons, x-rays and delta-rays, (*ii*) limiting the emission angle to a certain range, (*iii*) replacing the disintegration scheme by a source of monoenergetic gammas, (*iv*) parametrizing replicas for the collimator hole arrays (SPECT), (*v*) compressing voxelized phantom, (*vi*) techniques such as variance reduction, bootstrapping and jackknifing, (*vii*) splitting the simulation on a cluster, and recently on the grid [10].

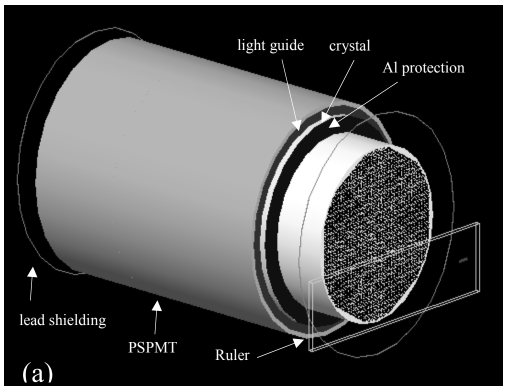
	Total		GATE availability (max)		GATE availability (average)	
	Clusters	CPUs	Clusters	CPUs	Clusters	CPUs
Day	11	886	7	430	7	125
Nights and Weekends	11	866	7	555	7	215

Table 1 Number of clusters and CPUs available to use GATE on CiGri.

Nowadays, the 'gridified' version of GATE is adopted by a growing number of users. The success of this method is related to the fact that the user is able to generate a large number of events using voxelized geometries and to obtain results in a reasonable time. To distribute a long sub-divided simulation on multiple processors, one should also sub-divide the associated long RNS into short ones. However, these short RNS have to be independent because any intra or inter-sequence correlation could lead to inaccurate results. The parallelization of the RNS was accomplished by using the sequence splitting method (also known as the random spacing method). The pseudo random numbers generator (PRNG) James Random (period 2^{144}) implemented by default in GATE was replaced by the Mersenne Twister PRNG due to his huge period (2^{19937}) [11]. The work described in this paper was done by using GATE 3.1.1 built upon Geant4.8.1. Table 1 shows the number of clusters/CPUs on which GATE was installed. The GATE macro contains a unique random number status and an output filename. These two parameters are renamed according to their appropriate value during the distribution process. Output files were analyzed with the ROOT object oriented data analysis framework.

2.3 The γ Imager: system configuration

The γ Imager is a high resolution planar scintigraphic camera which associates a 4 mm thick NaI(Tl) crystal and a PSPMT Hamamatsu R3292 which leads to a circular 100 mm diameter field of view. While most high resolution cameras use a pixilated crystal, the scintillation crystal of the γ Imager has a continuous 120 mm diameter with a 100 mm circular diameter active area. An aluminium protection layer, 0.8 mm thick, is placed in front of the crystal. The 5 Inch diameter PSPMT is equipped with a bialkali photocathode, 11 multiplication dynodes, 1 reflective dynode and a multiwire anode with 28 (X) + 28 (Y) wires. The readout of the 56 anode signals enables to calculate the spatial position (X, Y) and the energy for each event. A removable low-energy high-resolution parallel hole collimator with 35 mm thickness is used. The flat-to-flat distance of the hexagonal holes is 1.3 mm and the septum thickness in all directions is 0.2 mm. The whole detection head is surrounded with a 15 mm thick lead shielding. The γ Imager consists of two detector heads, only one head was used in this study.



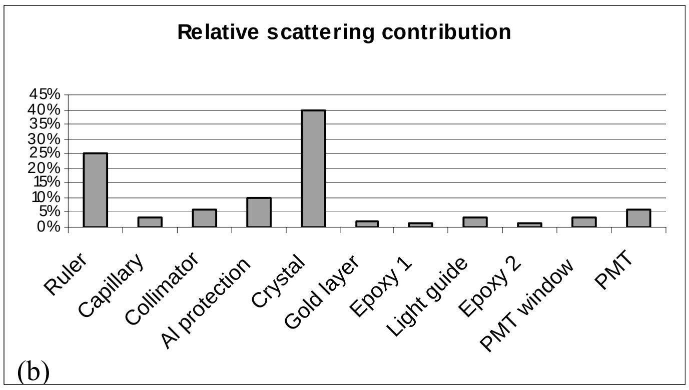


Figure 1 (a) The Biospace small animal camera modelled with GATE, (b) contribution of the scattered photons in each component of the simulation where the source is placed at 2 cm from the collimator.

2.4 Simulation of the Biospace gamma camera

The geometry of the γ Imager was accurately described in GATE. Figure 1 (a) shows the model of the detector head. The dimensions and the material of each part of the real camera were modelled as realistically as possible. The detector head simulation was thus composed of the lead collimator, a 1.2 mm air gap space, the aluminium protection, the NaI(Tl) crystal and the lead shielding of the whole head. As the plate scintillation crystal is 4 mm thick, about 38% of the 140 keV incident photons pass through the crystal without interacting. Thus, a significant number of photons will not be detected if we do not take into account the back-compartment. A 10 µm thick gold layer is placed behind the crystal, followed by a 0.39 mm Epoxy layer which separates it from the 1 mm light guide made of quartz. Another 0.15 thick Epoxy layer is positioned behind the light guide followed by the PSPMT, modelled as a 2 mm borosilicate glass entrance window and a 110 mm nickel backpart. The back-compartment of the detector was ended by a 15 mm rear lead shielding. Figure 1 (b) illustrates the relative scattering contribution of each layer in the model (source at 2 cm from the collimator), by counting the scattered events in each particular part divided by the total number of scatter events. During the acquisitions, a solution containing ^{99m}Tc, a photon emitter at 140 keV was put in a glass capillary of 1.4 mm inner diameter and 1.8 mm outer diameter and 6 mm length. The capillary was held by a Pyrex Ruler (150 mm x 40 mm x 3 mm). The Ruler and the capillary were also simulated.

2.5 Validation of the small animal camera

The validation of the small animal SPECT camera is based on the comparison of four parameters measured experimentally with the corresponding simulation data ; Energy Spectra, Spatial Resolution (FWHM), Sensitivity and Image of a capillary phantom. The three first parameters represent three basic features of a gamma camera, while the image of an inhomogeneous phantom allows a visual comparison between experimental and simulated data.

Experimental set-up

In this study, 16 planar acquisitions were made altogether. The radioactive background was first measured during 30 minutes without any activity and was subtracted from all the other measurements after normalizing it to the same acquisition duration. Then, 14 acquisitions were performed using a liquid source of ^{99m}Tc (140 keV) which was put in a thin capillary (1.4 mm inner diameter and 6 mm long) which was considered as a point source. Total activity was 70 µCi (2.59 MBq). The capillary was first placed in the air at 2 cm, 7 cm, 10 cm and 16.5 cm from the detector's surface. Then, the capillary at 16.5 cm was positioned above a cylindrical beaker (10.5 cm inner diameter and 15.6 cm long) which was put on the collimator. The beaker was used three times: empty, filled with 400 ml (46.2 mm height) and filled with 1000 ml (115.5 mm height) of water. All the seven configurations were performed in a first set of measurements with the source at the center of the camera Field Of View (FOV), and a second set was performed with the source 2 cm off-centered. The sixteenth configuration was performed so as to evaluate the image of the capillary phantom. The phantom consisted of four parallel capillaries (1.4 mm inner diameter and 31.5 mm long), with a capillary-to-capillary distance of 5 mm. The capillaries were filled with ^{99m}Tc solutions of different activities (from the least (right) to the most (left) radioactive) (cf. Figure 5 (a) and (b)): 81 µCi, 129 µCi, 220 µCi and 611 µCi. The phantom was 15 mm far from the scintillation camera. Events were recorded in an energy window between 40 and 186 keV for a duration of 5 minutes for each acquisition. The size of the projections was 256 x 256 pixels (pixel size ~ 0.39 mm).

GATE simulations

The fifteen configurations mentioned above were accurately reproduced using GATE. Monoenergetic gamma rays (140 keV) were emitted in 4 π steradians. The physical processes involving photon interactions (photoelectric effect, Compton scattering and Rayleigh scattering) were modelled using the low-energy electromagnetic package of GEANT4, while gamma-conversion was disabled. The energy resolution Full Width Half Maximum (FWHMe) of the camera was modelled by the convolution of the output data using a Gaussian distribution with user-defined mean and FWHM as stated by the manufacturer (11.5% FWHMe at 140 keV). The intrinsic spatial resolution FWHMi was also modelled in the same way (2.2 mm FWHMi at 140 keV).

Parallelization of the simulations

Many models of the camera have been tested in order to obtain the most realistic detector. A big simulation of one billion events was generated for each configuration. The simulation was split into 1000 small simulations, 1 million emitted events each, and was then distributed on CiGri. The 1000 small output files were collected from the grid, merged into one file (on a local CPU) and finally analysed using ROOT. Simulating 1 million events takes 10 minutes on a local CPU (Pentium IV, 3.2 GHz, 1 Go RAM) and requires ~ 30 million random numbers. Thus, a global sequence of more than 30 billion PRN was used for each simulation (which is small for the used Mersenne Twister).

Comparison parameters

Experimental and simulated data were compared basing them on the following quantities:

• *Energy Spectra.* The energy spectra of events recorded in the whole FOV (40-186 keV), was sketched for five configurations: (*i*) in air: source placed 2 and 10 cm from the collimator, (*ii*) in water: source placed at 16.5 cm from the collimator above the empty beaker which then is filled with 4.62 cm and 11.55 cm of water.

• *Spatial Resolution.* The FWHM of the point spread function (PSF) was calculated for the 14 configurations by drawing a 6 pixels thick profile (~ 2.345 mm) through the point source. Events were detected in the photopic window (126-154 keV) to reduce the noise effects.

• *Sensitivity*. The system sensitivity, defined as the number of detected events divided by the number of emitted events, was evaluated for the 7 configurations where the source is placed in the center of the FOV. Data were acquired in 3 energy windows: (*i*) photopic window 126-154 keV, (*ii*) Compton window 92-125 keV and (*iii*) total Field Of View (FOV) window 40-186 keV.

• *Image of a capillary phantom*. The image of the capillary phantom was acquired in the whole FOV in the energy window 40-186 keV. A 6 pixels thick profile was also drawn through the 4 line sources seen on the image.

3 Results and discussions

3.1 Energy Spectra

Figure 2 shows the experimental and simulated energy spectra of the ^{99m}Tc point source placed in the Air at 2 and 10 cm from the collimator: contributions of the simulated photons scattered within different components of the detector head and within the capillary and the ruler are also plotted. The experimental and simulated energy spectra were normalized to the same number of counts detected at 140 keV. As illustrated in figure 1 (b), including a back-compartment model was essential as well as the simulation of the capillary and the ruler to obtain a good agreement between measured and simulated spectra between 80 and 120 keV. The contribution of the scattered photons within the capillary and the ruler is clearly higher in figure 2 (a) (source at 2 cm) than in figure 2 (b) (source at 10 cm) especially between 70 and 80 keV. The smaller the ruler-collimator distance, the more the detected back-scattered photons in the ruler.

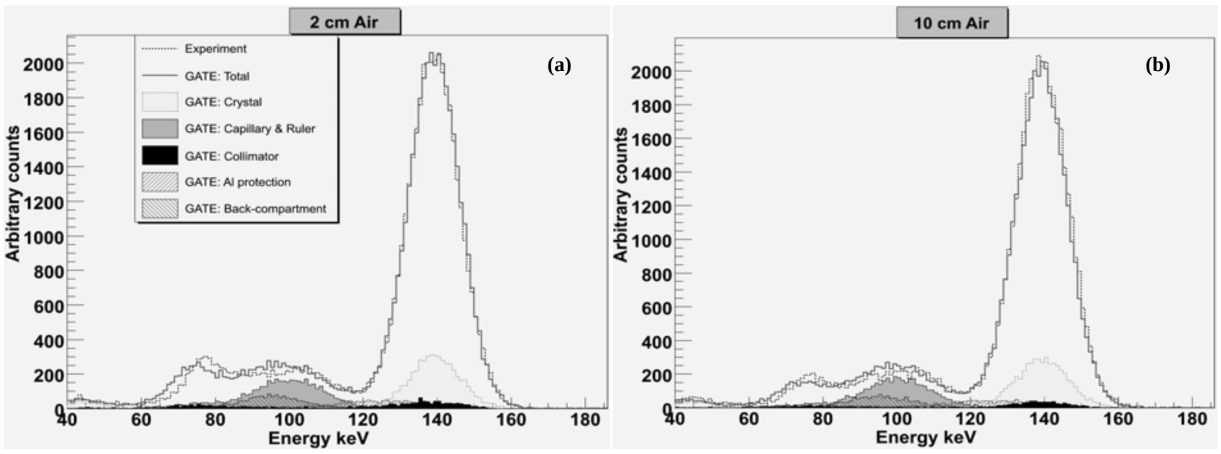


Figure 2 Experimental and simulated energy spectra for a ^{99m}Tc source positioned in Air at (a) 2 cm and (b) 10 cm from the camera.

Good agreements between experimental and simulated energy spectra can be noticed especially in the photopic and Compton window. Differences could be observed in figure 2 (a) between 60 and 100 keV. These differences are relatively the same between 90 and 100 keV in figure 2 (b) but decrease between 60 and 90 keV where the ruler has the biggest impact. Thus, the difference between 60 and 90 keV could come from the imperfect estimated shape and material of the ruler. For the other difference between 90 and 100 keV, the ruler and the back-compartment have the most important influence according to the figures. It couldn't derive from the ruler because this difference is relatively the same in figure 2 (a) and (b). Otherwise, the number of scattered photons would increase for the source at 2 cm from the collimator. Thus, the difference between 90 and 100 keV could be the consequence of the unawareness of a layer which was not modelled behind the crystal. Energy spectra obtained for the ^{99m}Tc point source at 16.5 cm from the collimator, with 0, 4.62 (400 ml) and 11.55 cm (1000 ml) water thickness are shown in figure 3. The scattered photons contributions were also drawn. The scattered photons within the phantom have increased, as expected, proportionally to the water thickness. Good agreements between experimental and simulated energy spectra could be observed for the two configurations with the empty beaker which then is filled with 4.62 cm thick of water. The same differences with the source in Air between 60 and 100 keV were maintained in figures 3 (a) and (b), which confirms that a back-compartment component was not modelled. Significant fluctuations were displayed in figures 3 (b) and (c) because of the low detected events in the corresponding images. This could explain the difference at the top of the photopic in figure 3 (c). The disagreement in figure 3 (c) between 70 and 80 keV and between 110 and 120 keV could be the result of the impurity of the real water used in the measurements.

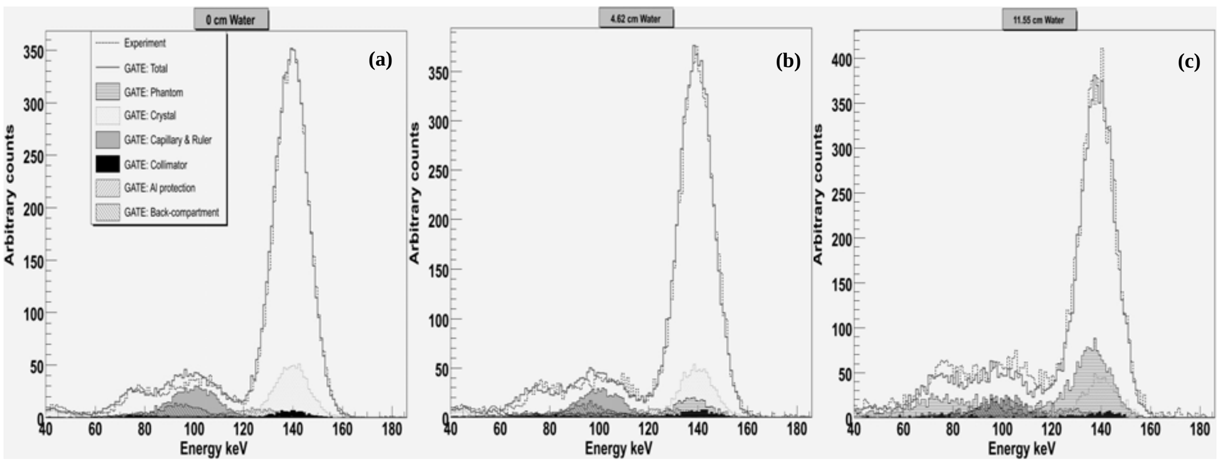


Figure 3 Experimental and simulated energy spectra for a ^{99m}Tc source positioned at 16.5 cm from the collimator above a beaker filled with (a) 0 cm, (b) 4.62 cm and (c) 11.55 cm of water.

3.2 Spatial Resolution

Distance source-collimator (water thickness)	Centered Source			Off-centered Source		
	Experimental FWHM (mm)	Simulated FWHM (mm)	Difference (%)	Experimental FWHM (mm)	Simulated FWHM (mm)	Difference (%)
2 cm	3.34	3.22	3.64	3.58	3.20	10.5
7 cm	4.53	4.54	1.89	4.75	4.53	4.49
10 cm	5.32	5.40	1.47	5.61	5.39	3.95
16.5 cm	7.25	7.34	1.30	7.64	7.37	3.49
16.5 cm (0 cm water)	7.27	7.39	1.63	7.63	7.38	3.32
16.5 cm (4.62 cm water)	7.26	7.32	0.91	7.62	7.26	4.69
16.5 cm (11.55 cm water)	7.58	7.65	0.91	7.83	7.61	2.85

Table 2 Comparison between experimental and simulated spatial resolution for the centered and off-centered sources placed at: 2, 7, 10 cm and 16.5 in air; 16.5 cm with three water thickness 0, 4.62 and 11.55 cm.

Table 2 shows the experimental and simulated FWHM of the PSF respectively for the centered and off-centered sources positioned in the Air at 2, 7, 10 and 16.5 cm from the collimator. Experimental and simulated FWHM of the PSF for the centered and off-centered sources placed above the phantom (0, 4.62 and 11.55 cm Water thickness) at 16.5 cm from the collimator are also shown. Experimental FWHM for a centered source were well reproduced by the simulations. The disagreements between experimental and simulated FWHM for an off-centered source is lower than 5%, except the source at 2 cm. The 5% differences might be due to the imperfect modelling of the PSPMT non-uniform response in GATE. The approximate modelling of the ruler was also noticed as the disagreements of the FWHM for the source centered and off-centered at 2 cm is clearly higher than the other configurations. An excellent agreement between experiment and simulated FWHM can be observed for the centered sources in water and a disagreement within 5% is obtained for the off-centered sources.

3.3 Sensitivity

The Biospace small animal camera sensitivity values obtained in Air in three energy windows (total FOV 40-186 keV, photopic window 126-154 keV and Compton window 92-125 keV) with GATE compared to the measured values are plotted in Figure 4 (a) for the 4 source-collimator distances 2, 7, 10 and 16.5 cm. Relative differences between experimental and calculated values were respectively: (*i*) 40-186 keV: 6.8%, 4.6%, 4.1% and 5.7%; (*ii*) 126-154 keV: 7.9%, 6%, 5.9% and 7.2%; (*iii*) 92-125 keV: 8.1%, 4%, 1.4% and 5.9%. The results of the system sensitivity with the phantom (empty, 400 ml and 1000 ml) derived from the experiment measurements and the simulations are drawn in figure 4 (b) for the source placed at 16.5 cm from the collimator where events were detected in three energy windows (Total FOV 40-186 keV, photopic window 126-154 keV and Compton window 92-125 keV). Relative differences between experimental and simulated values for the different water thickness (0, 4.62 and 11.55 cm) were: (*i*) 40-186 keV: 1.5%, 3.8% and 3.1%; (*ii*) 126-154 keV: 3.2%, 4.4% and 1.9%; (*iii*) 92-125 keV: 0.3%, 0.5% and 5.6%. The simulation was unable to reproduce the system sensitivity very accurately for the seven configurations described above. This difference could result from the inhomogeneous materials of the different components of the detector which only the constructor accurately knows.

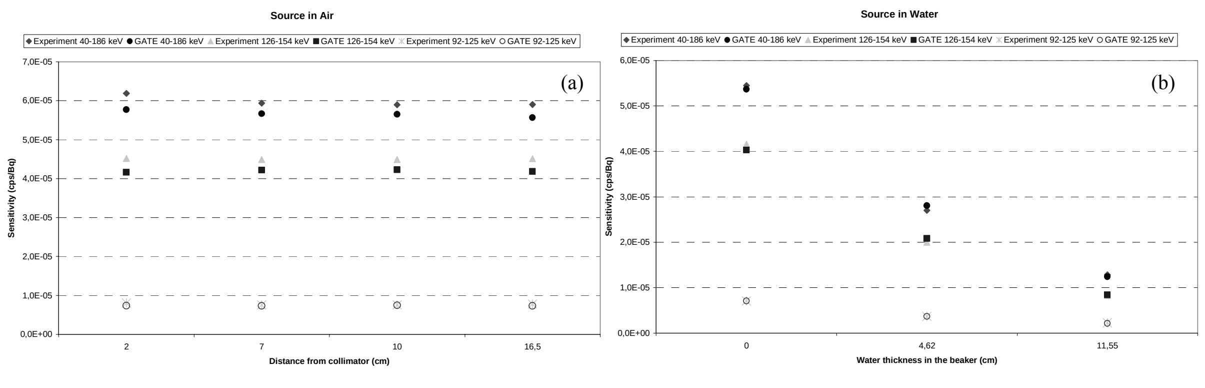


Figure 4 Comparison between experimental and simulated system sensitivities in three energy windows for the centered source placed at: 2, 7, 10 cm and 16.5 in air; 16.5 cm with three water thickness 0, 4.62 and 11.55 cm.

3.4 Image of a capillary phantom

Figure 5 shows a visual comparison between experimental (a) and the simulated (b) images of the capillary phantom as well as the horizontally drawn profiles (6 pixels thick ~ 2.345 mm) through these images (figure 5 (c)). The comparison of the two images and profiles shows a good agreement. However, the simulation was unable to perfectly reproduce the local distortion which could account for the 5% differences between the experimental and the simulated FWHM for an off-centered source.

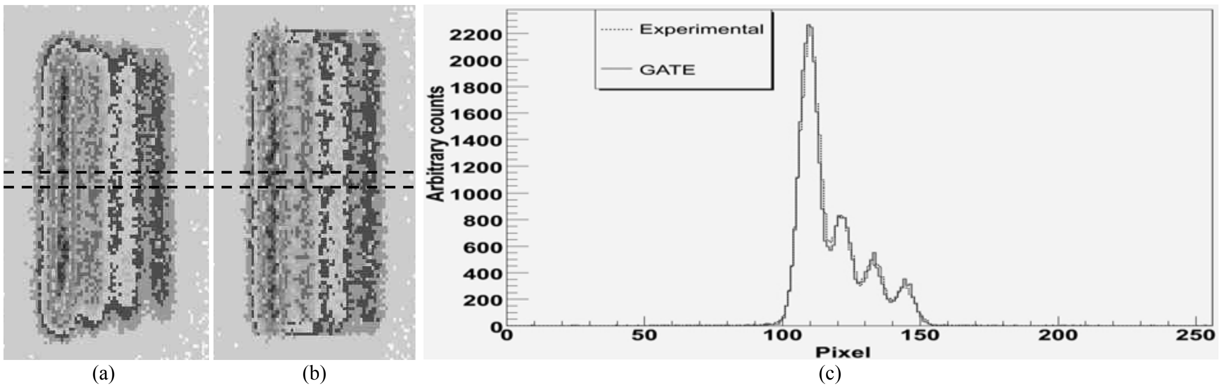


Figure 5 Comparison between experimental (a) and simulated (b) image of a four capillaries phantom filled with different ^{99m}Tc concentrations and horizontal profiles through the middle of these images (c).

3.5 Calculation time

Table 3 contains the computation time of a long simulation (1000 jobs) on a local CPU (Pentium IV, 3.2 GHz, 1 Go RAM) and on CiGri: during the day, during nights and weekends and an estimated average of the CPUs availability. CiGri has reduced the duration of a simulation compared to 1 CPU computation by a factor of respectively 42, 67 and 56. Resubmitted jobs rate on CiGri, defined as the number of resubmitted jobs divided by the number of executed jobs, is also represented in the box at the right of table 3. 16.9% of the jobs were killed and then were resubmitted by CiGri during the day which is as expected higher than the 10.2% resubmission rate during the nights and weekends. The estimated average of the resubmitted jobs is 13.4%. Each long simulation corresponds to one configuration. The material composition and the thickness of many components of the detector weren't exactly known (PSPMT, Gold layer…). In consequence, about 200 simulations were carried out in order to optimise the model of the Biospace camera. All these simulations would have taken 4 years of computation time on a local CPU, but on CiGri it took only 25 days (table 3). In the ideal case, where no job is killed, the 200 simulations would have taken about 22 days on CiGri. Thus, the 200 simulations would not have been feasible on a local CPU. Reducing the number of simulations would have altered the quality of the results. So the deployment of the Grid was crucial in this study to get accurate results.

	1 simulation (1000 jobs)	200 simulations	Gain	Resubmission percentage (%)
Local CPU	167 h	1392 days (~ 4 years)	1	0
CiGri – day	4 h	37 days	42	16.9
CiGri – nights and weekends	2.5 h	21 days	67	10.2
CiGri – estimated average	3 h	25 days	56	13.4

Table 3 Comparison between computing time of 1 and 200 simulations on a local CPU and on the CiGri Grid with the percentage of the resubmitted jobs (in the box at the right).

4 Conclusions and perspectives

Nuclear medical imaging needs a better physical model to improve the quality of the reconstructed images. Our study has brought into light that the Monte Carlo simulations toolkit GATE was able to accurately model the Biospace small field of view γ Imager. The camera model can thus be used for future researches such as developing new attenuation and diffusion correction algorithms. This paper has also proved the interest of grids in order to obtain accurate results within reasonable elapsed time. CIMENT GRID was an essential tool to carry this study to a successful conclusion. One of our coming studies will be the development of an iterative reconstruction algorithm in which long GATE simulations should be carried out at each iteration. For such studies, a higher scale of computation power is needed. The EGEE (Enabling Grid for E-sciencE) [12] Grid with a next multi-core processor generation should be able to meet our requests.

References

[1] A. Del Guerra and N. Belcari. *State-of-the-art of PET, SPECT and CT for small animal imaging.* Nucl. Instr. and Meth., A 583 pp 119-124, 2007.

[2] D. Lazaro et al. *Validation of the GATE Monte Carlo simulation platform for modelling a CsI(Tl) scintillation camera dedicated to small-animal imaging*. Phys. Med. Biol., 49 pp 271-285, 2004.

[3] F. J. Beekman and B. Vastenhouw. *Design and simulation of a high-resolution stationary SPECT system for small animals.* Phys. Med. Biol., 49 pp 4579-4592, 2004.

[4] C. Merheb et al. *Full modelling of the MOSAIC animal PET system based on the GATE Monte Carlo simulation code*. Phys. Med. Biol., 52 pp 563-576, 2007.

[5] I. Buvat and D. Lazaro. *Monte Carlo simulations in emission tomography and GATE: An overview*. Nucl. Instr. and Meth., A 569 pp 323-329, 2006.

[6] V. Breton, R. Medina and J. Montagnat. *DataGrid, Prototype of a Biomedical Grid*. Meth. Inf. Med., 42(2) pp 143-147, 2003.

[7] Ciment and CiGri project: https://ciment.imag.fr/cigri

[8] OAR: http://oar.imag.fr/

[9] S. Jan et al. *GATE: a simulation toolkit for PET and SPECT*. Phys. Med. Biol., 49 pp 4543-4561, 2004.

[10] L. Maigne et al. *Parallelization of Monte Carlo simulations and submission to a grid environment*. Parallel Process. Lett., 2004.

[11] R. Reuillon, D. R. C. Hill, Z. El Bitar and V. Breton. *Rigorous Distribution of Stochastic Simulations Using the DistMe Toolkit*. IEEE Trans. Nucl. Sci., 55(1) pp 595-603, 2008.

[12] EGEE Project: http://public.eu-egee.org/

Towards a Virtual Radiological Platform Based on a Grid Infrastructure

Sorina Camarasu[1], Hugues Benoit-Cattin[1], Laurent Guigues[1], Patrick Clarysse[1], Olivier Bernard[1], Denis Friboulet[1]

[1]CREATIS-LRMN,
CNRS UMR5220, INSERM U630, Claude Bernard Lyon 1 University, INSA Lyon,
Villeurbanne France

Abstract

The proposed Virtual Radiological Platform (VRP) project aims at providing realistic multi-modal medical images with 'ground-truth' knowledge. It relies on medical image simulators which often represent heterogeneous and computing demanding applications dealing with large amounts of medical images. In this context, computer grids are interesting architectures providing large computing power and a valuable collaborative framework. In this paper, the analysis of medical imaging applications gridification, based on several experiences, enables us to target key points for a contributory integration of grid architectures within a VRP framework.

Contents

1 **Overview of the Virtual Radiological Platform** 86

2 **Grid Contribution to VRP** 88

3 **Experience Feedback on Application Porting** 89

4 **Open Questions and Conclusion** 92

5 **Acknowledgements** 93

6 **Bibliography** 93

The paper begins with an overview of the Virtual Radiological Platform (VRP) defining its aim, usage and requirements. Chapter 2 briefly describes the two key elements of the grid contribution to such a VRP. Our experience feedback regarding medical application porting on grid infrastructures is summarized in a step-by step gridification approach in Chapter 3. Finally, we discuss problematic issues and draw the conclusions in Section 4.

1 Overview of the Virtual Radiological Platform

1.1 VRP Objective

As illustrated in Figure 1, the VRP aims at providing realistic medical images relying on multi-modal simulators and taking as an input a virtual model. Such a model is a parametric description of an object of interest (organs, body…) in accordance with the simulation process (proton density, diffusion absorption parameter etc.). The following modalities are targeted: Magnetic Resonance Imaging (MRI), Magnetic Resonance Spectroscopy (MRS), Ultrasound (US), XRay and Computed Tomography (CT), Positron Emission Tomography (PET) as well as Radiotherapy and Hadrontherapy.

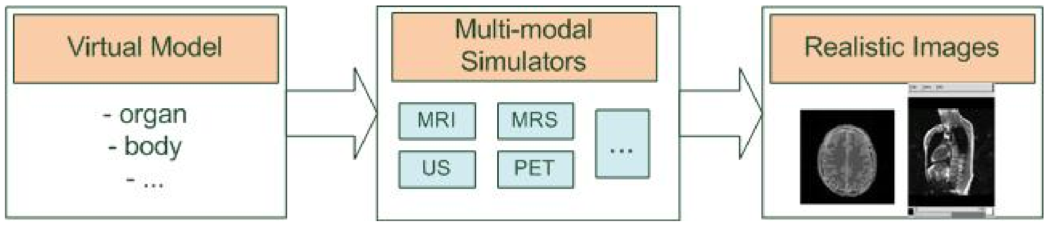


Figure 1 - VRP Overview.

Note that for these modalities a certain number of simulators have already been developed and used: *Simri* [1] which is a 3D MRI Simulator, *Field* [2] for the US simulation, *Gate* [3] and *Sorteo* [4] in the PET field and *ThIS* [5] for Hadrontherapy. In view of the VRP, recent tests have been performed using the specific simulation of MR and PET images of a breathing thorax and beating heart [6].

In a similar context, GEMSS (GRID-enabled Medical Simulation Services) [7] was a two and a half year project which started in 2002 and aimed at enhancing healthcare by bringing a variety of medical computing and resource services into the user's environment and providing access to advanced simulation and image processing services. The project led to the development of a grid middleware integrating a certain number of services. Their architecture was based on standard web services.

Note that GEMSS was mainly oriented towards developing an innovative middleware, whereas we are more oriented towards using existing grid middleware for building an architecture (VRP) allowing for a unified and user-friendly usage of multi-modal simulators.

1.2 VRP Usage

Simulators are particularly useful in the field of medical imaging, as they can serve different purposes. Figure 2 summarizes some of the main functionalities of the VRP:

• Treatment planning: for example *ThIS* [5] is a software dedicated to the Monte-Carlo simulation of irradiations of living tissues with photons, protons or light ions beams for cancer therapy planning [8].

• Theoretical understanding of physical phenomena: a MRI simulator like *Simri* [9] is naturally suited to acquire theoretical understanding of the complex MR technology. It can also be used as an educational tool in medical and technical environments.

• Acquisition Protocol Definition/Optimization: in the case of MRI for example, because of the complexity of the in-vivo MRI acquisitions, the MRI protocol definition and optimization are extremely difficult, but they become accessible through a simulator

• Assessment tool for image analysis methods: simulators produce realistic images that can be used to evaluate the results of image processing algorithms as the ‘ground truth’ is available through the virtual model of anatomical structures

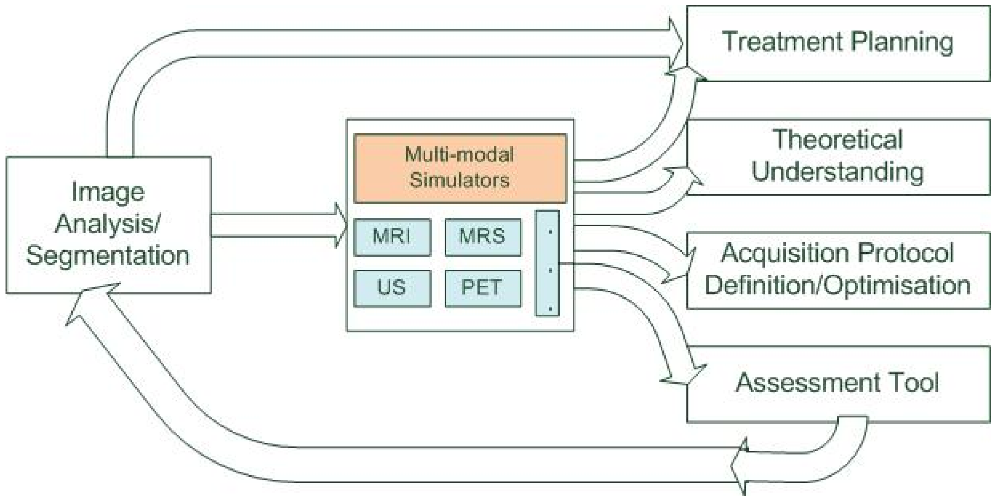

Treatment Planning
Multi-modal Simulators
Theoretical Understanding
Image Analysis/ Segmentation
MRI
MRS
US
PET
Acquisition Protocol Definition/Optimisation
Assessment Tool

Figure 2 - VRP Use Cases.

At a larger scale, we can imagine a complete simulator of a Virtual Physiological Human [10], i.e. a multi-scale human modeling for a preventive and predictive personalized health. This kind of project is only possible if existing heterogeneous medical imaging applications and especially simulators can be brought together and rendered interoperable on a same platform. Such high level objective is the long-term goal of the FP7 NoE VPH project.

1.3 VRP Requirements

Simulators interoperability and easy plug-in of new simulators are the main requirements for the VRP. In the last few years, a certain number of simulators have been used and developed inside the medical community. Their usage is unfortunately restricted because of their non-interoperability and of their heterogeneous requirements (dependencies, interface, parameters etc.).

Another important aim is making the simulators and the data accessible to everyone. Such a platform should allow users to run simulations without having to install them on their own computer.

Moreover, the platform should have access to distributed architectures providing important computing and storage resources. This will lead to a faster execution for time consuming applications and to a discharge of the user work station. Parallel and distributing computing will also ensure scaling capabilities, i.e. the possibility of supporting a large number of users and jobs simultaneously. However, the parallel and distributed computing platform has to be transparent to the end user, who does not need to acknowledge the complexity of the underlying infrastructure.

In the next section we will explain how grids can meet these VRP requirements.

2 Grid Contribution to VRP

As presented previously, the VRP has to be able to offer both a collaborative platform and computing resources for the medical image simulators execution. These two requirements can be met with the help of grid infrastructures exploited in various medical and non-medical projects. The next two paragraphs will illustrate this through the example of several medical imaging projects using grid infrastructures for building collaborative platforms and for intensive computing.

2.1 Example of collaborative platforms

In the context of breast cancer, the *MammoGrid* project [11] developed a distributed platform for a community of radiologists. They used the grid infrastructure in order to support collaborative medical image analysis, to enable radiologists share standardized mammograms, compare diagnosis etc. Thus different sites from Udine, Geneva, Cambridge and Oxford contributed at creating an international mammogram database like illustrated in Figure 3. In their implementation a lightweight Grid middleware solution (called *AliEn*) is used. They also developed a set of services called the MammoGrid Services (MGS) for managing mammography images and associated patient data on the grid [12].

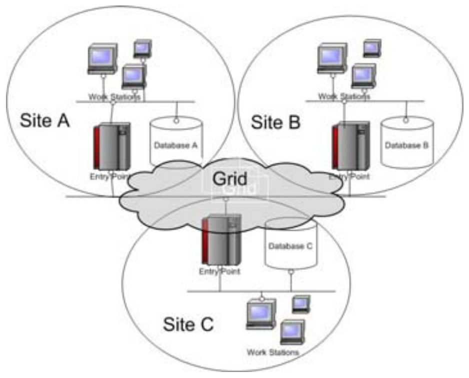


Figure 3 – Illustration of the distributed platform developed within the MammoGrid Project.

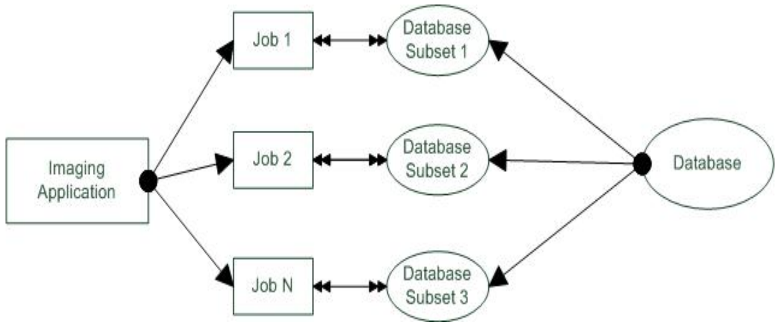


Figure 4 - Database and processing partitioning.

Still on breast cancer, the MAGIC-5 Project [13] also used a dedicated *AliEn* Server for their database architecture allowing for images to be acquired in any site available to the Project. Data are stored on local resources and recorded on a common service, known as Data Catalogue, together with the related information (metadata).

2.2 Intensive computing

Grids are also extremely valuable for their computing resources. In the case of a VRP, a certain number of medical imaging simulators are needed. Grid computing parallelism allows then for important speed-up for such computing intensive applications. We can refer to it as *processing partitioning*. As illustrated in Figure 4, processing partitioning can be coupled with *database partitioning* for higher efficiency of the medical applications which deal with large amounts of data (e.g. medical image indexing and retrieval).

If the database is partitioned in bags of images to be analyzed, each bag can be processed by a single computing job. If one bag represented one image so that all images were processed in parallel, then the execution time would be the maximum of the execution times of each image processing [14]. However, for large databases this hypothesis is not realistic, as thousands of processors cannot be available simultaneously on a shared grid infrastructure concurrently exploited by a large number of users.

Processing partitioning is studied in [14]. The strategy for choosing the granularity of the tasks submitted aims at reducing both the total execution time and the number of submitted sub-jobs (for infrastructure optimization). A probabilistic model is proposed and validated based on experimental results. A more detailed study on submission strategies optimization can be found in [15]. This paper studies the influence of various parameters as for example execution site, resource broker or day of the week. It shows that the grid latency is closely related to the choice of a resource broker or a computing element and that information concerning these two parameters may be both sufficient and accessible for job submission and system performances optimization.

3 EXPERIENCE FEEDBACK ON APPLICATION PORTING

A large number of medical imaging applications including simulators have been implemented without having taken into account a possible future grid usage. However, we have nowadays the possibility of using distributed architectures for these applications if we succeed in adapting them for grid usage, which comes within the scope of the VRP. This section summarizes our experience feedback regarding medical application porting on grid infrastructures in a step-by step gridification methodology. This methodology will also serve for future work needed for integrating new applications into our grid oriented VRP.

After several years of experience with porting applications on distributed architectures (starting with the European DataGrid Project in 2001), we can say that the gridification process requires several layers of adaptability:

3.1 Basic adaptability – successful execution

The application has to be able to run on the grid environment, i.e. operating system, file system, shared resources etc. The configuration of the distant grid nodes is often very little known to the grid user and does not necessarily correspond to the user's local configuration. Therefore, the basic adaptability can refer both to the distant grid node environment and to the application to be executed on it.

The distant environment often needs a certain 'customization' before job execution. The customization can vary from variable definition to file downloading from accessible sources. These operations can be done through a shell script that launches the application after the necessary environment customization. Depending on the used platform and on the user's access rights, there will be a certain number of restrictions that may not allow for all the changes the users would like to bring in order to run his/her application. In this case, he will have to customize the application.

This may be the case for applications requiring a certain number of libraries that are not present on the distant nodes. *ThIS* [5] for example is based on the *Geant4* [16] toolkit for the simulation of the passage of particles through matter. *Geant4* is used in different application areas, mainly in high energy, nuclear and accelerator physics, but also in medical and space sciences. *ThIS* uses a certain number of the *Geant4* libraries which may be installed on some of the grid nodes, but not necessarily on all of them. Moreover, *Geant4* can build shared libraries to which *ThIS* can be linked. The solution we adopted in order to achieve a basic adaptability for *ThIS* was to re-build all dependencies without shared libraries and then compile *ThIS* by linking it statically to the required libraries.

Another possible solution is to copy the shared libraries on the nodes together with the executable or to create the necessary environment for compiling our application on the target execution node.

3.2 Intermediate adaptability – application parallelization

Grids are particularly efficient for intensive computing applications that can be parallelized. This is often the case for medical image applications. However, parallelization is sometimes not taken into account at the conception phase of the application. We will consider the example of *Simri* and *ThIS* which are two simulators parallelized in two different ways. *Simri* uses an MPI (Message Passing Interface) implementation, whereas *ThIS* deals with Monte-Carlo simulations.

MPI parallelization is transparent to the end user: task parallelization and message passing between the tasks executing on different processors is handled automatically if the application has been developed using one of the MPI implementations like for example LAM/MPI or MPICH. MPI is often used on computer clusters, where users have access to multiple processors simultaneously. *Simri* is a typical example of application that has been modified in order to be parallelized with LAM/MPI [17]. It can now be executed on a single PC, as well as on parallel machines, on a small internal cluster and even on larger infrastructures like the EGEE Grid.

The experience of using both infrastructures for *Simri* showed that MPI is more adapted for cluster usage than for grid usage. A first issue when using grids concerns the previous adaptability point: the different computing elements which can be distributed geographically all over the world and have different administrators may not support the needed MPI implementation. A second issue is that usually the same application cannot be split between different computing sites. This means its parallelization is limited to one structure and cannot benefit from the free resources elsewhere. This limitation could be overcome with wrappers and we can then speak of a higher layer of adaptability.

ThIS is a *Geant4* based software dedicated to the Monte-Carlo (MC) simulation of irradiations of living tissues with photons, protons or light ions beams for cancer therapy. For reliable results, a large number of particles are simulated, leading to intensive computing. However, this large number of particles needed for one complete simulation can be split into sub-simulations with fewer particles which are thus less computing intensive. The condition to observe when splitting one simulation is that each sub-job must use a different random seed number. This allows the sub-simulations to be statistically independent and to be thus run concurrently. The parallelization in this case is achieved naturally by simulating fewer particles concurrently on multiple processors. For example, if we want to simulate 50 M particles, the simulation is divided into 50 sub-jobs, each with 1 M particles to simulate. However, additional tools are needed for job splitting and result merging.

In conclusion, the parallelization technique depends essentially on the application. For independent MC particle simulations the splitting and merging are straightforward and therefore they can be done with external generic tools as presented in the next section. On the contrary, in the case of the MRI simulation splitting and merging requires image partitioning and reconstruction which cannot be achieved easily with the same tools. MPI parallelization solves this problem and renders the splitting and merging process transparent to the user, which explains our choice for the parallelization technique. Even if MPI may not have been conceived for usage on grid architectures, in practice a strong MPI demand has been observed within the biomedical community of the EGEE project. Their experience shows that for a large number of biological applications MPI is the most accessible parallelization technique, allowing for the code to be executed without any further effort on different architectures: parallel machines, clusters and grids.

3.3 Advanced adaptability – wrappers/descriptors

Advanced adaptability is meant to help the user take full advantage of the grid resources and application parallelism by bringing a certain degree of automation in the process of job parallelization and result retrieval. This can be achieved with additional tools like wrappers or even new software layers.

3.3.1 Parallel job submission, monitoring and retrieval

The previous section presented two means allowing for applications to be executed on multiple processors concurrently. However, both have their own limitations that can be overcome with additional tools.

In the case of MPI, the different processes of a same job could only execute on the processors of a same cluster. One of the topics covered by the Interactive European Grid is advanced MPI usage on the *int.eu.grid* infrastructure [18]. In this field, they developed several tools in order to enhance workload management features for MPI, as well as a cross broker allowing for cross-site MPI.

A natural parallelization like the one used in the case of *ThIS* can benefit from automatic handling of job splitting and result merging. The usefulness of his approach has been acknowledged so that today there are several tools that have been designed to implement it. The Java Job Submission (JJS) [19] tool for example can be used for easier job submission, monitoring and result retrieval. It hides most of the grid complexity from the user and also allows for simultaneous submission of similar sub-jobs called parametric jobs. However it does not take into account actual job splitting and merging. *Ganga* [20] is another interesting tool which includes a job splitter and merger. Moreover, it allows users to interact with multiple and different computing backends in a very similar way through the *Ganga* interface.

Grid middleware integrates basic tools for job submission, monitoring and retrieval on the grid. On the EGEE grid infrastructure for example, the Workload Management System (WMS) is used. The WMS is a very useful tool; however it is often not sufficient. The WMS allows for job submission and monitoring ‘manually’ with command lines. This requires a large amount of work for status querying for every job, then manual retrieval and merging of results. Despite its simplicity, the WMS also has a certain number of advanced functionalities among which we can cite the ‘parametric job’ type that allows the automatic submission of a number of very similar jobs through one job description file. However, this advanced feature still requires further improvement on certain aspects like global job status. At present, for example, if one submits N jobs at a time with one job description file for parametric jobs and some of the jobs are done and the others fail, the global job status stays ‘Running’. In conclusion, more sophisticated tools like Ganga or JJS are often needed for more efficient job management.

3.3.2 Middleware compatibility

The multiple computing backend issue brings us to another important aspect that could be handled as an advanced adaptability layer: middleware interoperability, as different technologies not only co-exist but are also changing and evolving continuously. In our case for example, *Simri* has access to the EGEE grid deploying the gLite middleware and to an internal cluster deploying PBS. The same application can be executed on both platforms due to a home-made tool (a portal) that translates user requests into the right language and deals with job submission, monitoring and retrieval depending on the backend.

3.3.3 Integration into service platforms

Last but not least, advanced adaptability can also refer to wrapping applications for integration into a service platform. On a service oriented platform heterogeneous applications need to be presented in a normalized language. The GEMSS project mentions application descriptors [21]. An application descriptor is an XML document that provides meta-data about an existing application and is usually supplied by the service provider. The information provided in an application descriptor defines the semantics of the operations of a generic application service and enables the service framework to expose an application automatically as a Web service. An application descriptor usually specifies the input/output files, the script for initiating job execution, scripts for gathering status information, and a finish file, which indicates that a job has been finished.

3.4 End-user adaptability

On top of these adaptability layers, there is a user-adaptability layer which is a high level interface allowing users with no grid specific knowledge to use the available services. Grid architectures usually offer transparent access to resources to grid users but they lack a generic user- friendly interface at a final user level. This hinders their use by physicians and researchers who would need secure and easy-to-use portals for submitting their jobs.

This problem has been acknowledged by different communities who have already developed a certain number of portals such as Genius [22], GridSphere [23] etc. A few communities have developed their own portals for specific needs. The *Simri* portal [24] has been developed for example for the *Simri* simulator. This 3 layer architecture portal was developed in Java and PhP. More information on the technical implementation can be found in [25]. Experience shows that the development of a specific portal for a given application is not a tremendous task. However, creating and especially maintaining a new portal for every application rapidly becomes a very time consuming job. The challenge thus becomes conceiving a more generic tool, i.e. a portal easily re-configurable for similar applications.

4 OPEN QUESTIONS AND CONCLUSION

Grids are promising architectures that can bring different solutions to medical image storage and intensive computing problems. Their growing interest is reflected through numerous projects and grids initiatives emerging worldwide. However, grid issues limiting the adoption of existing systems still exist.

In the previous section we tackled a step-by-step approach for medical application porting on grid architectures emerged from our own experience which shows that porting and deploying applications on grid infrastructures is not a straightforward process. Moreover, although this approach is meant to be general, different applications have different structures and requirements making it impossible to have a recipe 100% valid for all applications. The difficulty of the task is therefore increased.

For example, one problem that we still find difficult to overcome and that could be considered as an advanced adaptability issue concerns large input and output file management. Most grid middleware incorporate file catalogues and data management services, but job parallelization often requires multiple copies of data from storage elements/databases to the working nodes. This problem is particularly important for medical applications.

Last but not least, we noticed that an important element which seems to significantly hinder the growth of the grid user community is a higher user interface. In order to really render a gridified application accessible in practice one has to invest important resources for the user interface and support issues.

The VRP architecture should therefore take into account all these problems and facilitate the integration of medical simulators into the grid environment. Among the main requirements of the VRP platform we can cite the access to grid architectures and distributed data, easy plug-in of new simulators and personalized models, as well as a user-friendly interface. Designing and implementing the VRP's architecture represents thus a very challenging project. At present our team works on designing the first draft of the VRP architecture, which relies on Web services and the XML language.

5 ACKNOWLEDGEMENTS

This research work is in the scope of the scientific topics of the FP7 NoE VPH and of the French National Grid Institute (IdG). This work has been partially funded by the EGEE2 project within the European FP6.

Many thanks to David Sarrut for his collaboration on porting ThIS on the EGEE Grid, to Thomas Grimaldi for his contribution on the VRP project and to Fabrice Bellet for his help with application compiling and customization for the grid.

6 BIBLIOGRAPHY

[1] H. Benoit-Cattin, G. Collewet, B. Belaroussi, H. Saint-Jalmes, and C. Odet, "The SIMRI project: A versatile and interactive MRI simulator," *Journal of Magnetic Resonance*, vol. 173, pp. 97-115, 2005.

[2] J. A. Jensen, "Field: A Program for Simulating Ultrasound Systems," *Medical & Biological Engineering & Computer*, vol. 34, pp. 351-353, 1996.

[3] S. Jan, et al., "GATE: a simulation toolkit for PET and SPECT," *Physics in Medicine and Biology*, vol. 49, pp. 4543-4561, 2004.

[4] A. Reilhac, et al., "PET-SORTEO: a Monte Carlo-based Simulator with high count rate capabilities," *IEEE Transactions on Nuclear Science*, vol. 51, pp. 46- 52, 2004.

[5] ThIS Homepage, http://www.creatis.insa-lyon.fr/rio/ThIS

[6] R. Haddad, I. E. Magnin, and P. Clarysse, "A new fully-digital anthropomorphic and dynamic thorax/heart model," in *29th Annual International Conference of the IEEE EMBS*, Lyon, France, 2007, pp. 5999-6002.

[7] The GEMSS Project: Grid-enabled medical simulation services. EU IST Project IST-2001-37153, 2002--2005, http://www.gemss.de/

[8] A. GÓMEZ, et al., "Monte Carlo Verification of IMRT treatment plans on Grid," in *Healthgrid*, Geneva, 2007, pp. 105-114.

[9] Simri Homepage, http://www.simri.org/

[10] Towards Virtual Physiological Human: Multilevel Modelling and Simulation of the Human Anatomy and Physiology – White Paper, November 2005, http://europa.eu.int/information_society/activities/health/docs/events/barcelona2005/ec-vph-white-paper2005nov.pdf

[11] R. Warren, et al., "MammoGrid - a prototype distributed mammographic database for Europe," *Clinical Radiology* 2007.

[12] F. Estrella, et al., "Experiences of engineering Grid-based medical software," *International journal of medical informatics*, pp. 621-632, 2007.

[13] R. Bellotti, et al., "Distributed medical images analysis on a Grid infrastructure," *Future Generation Computer Systems*, 2007.

[14] J. Montagnat, T. Glatard, and D. Lingrand, "Texture-based Medical Image Indexing and Retrieval on Grids," *Medical Imaging Technology*, vol. 25, 2007.

[15] T. Glatard, D. Lingrand, J. Montagnat, and M. Riveill, "Impact of the execution context on Grid job performances" in *International Workshop on Context-Awareness and Mobility in Grid Computing (WCAMG'07)*, Rio de Janeiro, 2007.

[16] Geant4 Homepage, http://geant4.cern.ch/

[17] H. Benoit-Cattin, F. Bellet, J. Montagnat, and C. Odet, "Magnetic Resonance Imaging (MRI) simulation on a grid computing architecture" in *CCGRID* Tokyo, 2003.

[18] The Interactive European Grid Project, http://www.interactive-grid.eu/

[19] Java Job Submission Tool, http://cc.in2p3.fr/docenligne/269

[20] Ganga Homepage, http://ganga.web.cern.ch/ganga/

[21] S. Benkner, I. Brandic, G. Engelbrecht, and R. Schmidt, "VGE - A Service-Oriented Grid Environment for On-Demand Supercomputing" in *Fifth IEEE/ACM International Workshop on Grid Computing*, 2004.

[22] The Genius Portal, https://genius.ct.infn.it/

[23] The Gridsphere Portal Framework, http://www.gridsphere.org

[24] The Simri Portal, https://simri.creatis.insa-lyon.fr

[25] F. Bellet, I. Nistoreanu, C. Pera, and H. Benoit-Cattin, "Magnetic resonance imaging simulation on EGEE grid architecture: A web portal design." in *HealthGrid*, Valencia, 2006, pp. 34-42.

Issues in Managing Variability of Medical Imaging Grid Services

Mathieu Acher[1], Philippe Collet[1] and Philippe Lahire[1]

[1]Université de Nice Sophia Antipolis, laboratoire I3S - CNRS UMR 6070
Polytech Nice - Sophia, 930 Route des Colles, BP 145, F-06903 Sophia Antipolis Cedex, France

Abstract

In medical image analysis, there exist multifold applications to grids and service-oriented architectures are more and more used to implement such imaging applications. In this context, workflow and service architects have to face an important variability problem related both to the functional description of services, and to the numerous quality of service (QoS) dimensions that are to be considered. In this paper, we analyze such variability issues and establish the requirements of a *service product line*, which objective is to facilitate variability handling in the image processing chain.

Contents

1 Introduction **95**

2 An Analysis of Variability in Medical Imaging **97**
2.1 Segmentation . . . 97
2.2 Analysis of QoS Variability in Segmentation . . . 97
2.3 Impact of Variability for Grid Services . . . 98

3 Building the Service Product Line **99**
3.1 Software Product Lines . . . 99
3.2 Principles . . . 100
3.3 Open Issues . . . 102

4 Conclusion **103**

1 Introduction

For several years now the clinical area has been investigating grid computing to deal with many problems related to large medical data sets manipulation, usually heavily fragmented, on very wide distributed infrastructures. In medical image analysis, there exist multifold applications to grids, from validation and optimization processes of specific algorithms to overall reduction of computing time. In the same time, image analysis tool pipelines are undergoing homogenization, strongly motivated by the need for mutualizing software development and easily comparing results. A medical image processing library such as ITK or the statistical tool SPM are examples of these efforts. In both cases however, these approaches assume that codes are fully integrated and thus tightly coupled. Harnessing the full power of the grid implies to be able to deploy different variations of algorithms while being independent of the heterogeneity of existing codes. Moreover, these codes are often not limited to a single algorithm but are expressed as processing pipelines.

A service oriented architecture (SOA) is especially adapted to adress these requirements, with services inherently decoupled and abstracted from technical platforms, and workflows to design composed algorithms. It enables users to partially overcome the division of clinical centers and medical imaging laboratories. But building on SOA also implies to tackle two major problems in its usage on realistic use cases. First, code maintainers have to provide basic imaging services from heterogeneous codes, including detailed information enabling their composition to construct new pipelines and the control of their deployment on the grid. The second problem concerns the management of the numerous non functional properties that have to be exploited during deployment or run times, in order to ensure a quality of service (QoS) adapted to the user. These QoS properties expose different forms of variability as they may be related to a service itself (reliability, availability, cost, expected execution time...), to its provision on the grid (parallelism grain, data handling protocol, adaptability to resources...) or to some user needs (emergency of a computation, expected output quality...). The two problems put together lead to strong limitations in the application of SOA principles to the grid for medical image analysis. We identify these two problems as being related to an important variability in the service functionalities — there are similar basic services to choose from to build a complete workflow — and an even more important variability in the QoS related properties of the service — these properties can be about execution time, cost, quality of results, *etc* . —. This concept of variability is now an essential design elements in software engineering [2, 13]. Its realization can be seen as a way to describe the whole generality of an entity (a software artefact) through the specification of commonalities and differences. This approach allows software architects to i) describe the structure and the behaviour of a single entity, with the possible common parts that may exist between several similar entities; ii) propose variants of one entity and therefore specify possible variations within the structure and the functionalities that are provided; iii) define optional parts for both structural and behavioural aspects; iv) describe assembly and runtime constraints between entities (main types of constraints are mutual exclusion and dependancy constraints), and v) define behavioural variations that are implied by the specification of entity variants.

In our context, the management of variability concerns both the services and the resulting composed processes. Our long term goal is to provide a complete *software product line* [2] that describes major variations in functional and non functional properties of medical imaging services and workflows. A software product line applies the general industrial notion of engineering family of similar entities. The main focus is on ensuring or verifying appropriate properties on the instantiation of a single entity from the product line. As a start, our ongoing work is limited to a software product line handling only variability on the description of imaging services [7]. In this paper, we present the first analysis and specifications of such a software product line. We analyze the different aspects of variability in some medical imaging services, and we focus on relevant QoS properties, taking the segmentation services as a running example (Section 2). We then present the main principles of a software product line framework that would handle the identified variability points (Section 3). It notably has to enable service providers and workflow experts i) to capture the commonalities and the differences of legacy services, ii) to efficiently build the right service according to these commonalities and differences and iii) to use the line to select appropriate services according to functional and non functional criteria. We also describe how Model-Driven Engineering (MDE) techniques are used to build the service product line as a model of the manipulated services so that description, selection and compatibility checking can be made through the product line. Finally we discuss some identified open issues related to the implementation of the product line (Section 4).

2 An Analysis of Variability in Medical Imaging

Let us consider a general pipeline in medical imaging. For a same type of image processing there are most of the time several ways to build it and numerous services are available to the workflow experts for each step in the imaging process. Functional aspects supported by these services are highly variable: for instance, there are many algorithms for the segmentation stage that are designed for a specific image acquisition modality. Neverthless, the variability mostly concerns the *quality* offered by the service (QoS), which describes in which conditions functional aspects are provided. Indeed, a segmentation algorithm may perform very well for SPECT images when considering the accuracy criteria, while another one could be less accurate but much faster for the same input. These services may provide the same functionality, but optimize different QoS dimensions. It is thus not sufficient to only consider functional characteristics of services. We now analyze how and which QoS properties are computed and quantified in the medical image analysis area, considering one essential technique, namely segmentation.

2.1 Segmentation

The goal of segmentation is to select perceptual units of an image that correspond to the real anatomy of the patient, and which need to be measured or visualised by the clinical user. The process of segmentation is a crucial (and often preliminary) step for medical imaging analysis and diagnosis. Unfortunately, automatic segmentation is a problem without general solution, as segmentation results depend on many factors, such as modality acquisition, image noise level, organs / body region extracted, pathology, *etc*. Selecting an accurate and efficient segmentation technique can avoid or minimize inappropriate results. Consequently the need of a standard quality measure has been highlighted [20] in order to evaluate different qualities of segmentation algorithms.

Different evaluation methods have then been proposed and we identify them as a first degree of variability. First, the *analytical* methods directly consider principles and properties (such as requirements, utilities, complexity, *etc.*) of algorithms. Analytical methods have not received much attention, considering that they cannot obtain fine-grained properties, they only work in a particular context and they have difficulties to compare algorithms. Second, *goodness* methods compute image-specific properties of the segmented object such as intra-region uniformity, inter-region-contrast, entropy, shape, edge quality, *etc* . As pointed in [5], this approach implies the subjectivity of selected properties, but they do not require an explicit reference knowledge and they may give a fast evaluation in some cases. Finally, *discrepancy* methods rely on a gold standard. The availability of a reference segmentation, supposed to be an ideally segmented image, allows discrepancy to measure the agreement between a segmented object – the result of the segmentation process – and references. Similarity or difference measures between segmented and reference images are then computed.

2.2 Analysis of QoS Variability in Segmentation

We identify several variability points in QoS related to segmentation.

First evaluation methods have to specify the *application domain* under consideration, which according to [17] is determined by three entities: the goal of the segmentation (the task), the body region and the imaging protocol. For instance, a particular segmentation method may have high performance in determining the volume of a tumor in the brain on an MRI image, but may have a low performance in segmenting a cancerous mass from a mammography scan. It is thus necessary to introduce specific information about images and anatomical structures users want to identify, and this can be expressed through several choices (alternatives).

This survey of evaluation methods show that the QoS is context dependent: the absence of a reference image prevents using discrepancy methods, whereas the knowledge of both clinical context and medical objectives can largely improve QoS measurements. Statistical (or empirical) analysis, which considers mostly discrepancy methods defined above, must propose common criteria (*metrics*) to compare segmentation algorithms. But these metrics do not make any sense if they are not computed in a particular context. This is clearly expressed through constraints between alternatives.

Another important aspect is that the quality of the evaluation itself has to be considered [20]. Some evaluation methods focus on specific measures: validation metrics in medicine, such as sensitivity and specificity, have drawbacks not only for medical image processing but also for evaluation of medical tests. Moreover, the metrics selected are subjective or objective and this is another variation point. Complexity involved for evaluation may be important if, for instance, a monitoring process is used to control algorithms. Evaluation requirements are also considered. All those parameters lead to describing and handling complex interdependencies.

In many research works, the main quality considered is *accuracy* (aka fidelity), which refers to the degree to which the segmentation results agree with the “true” segmentation. We claim that the influence of various parameters cannot be measured accross this single quality dimension. It is more relevant, as proposed in [15][17] specifically for segmentation algorithms, to consider many dimensions to give a meaningful answer to the performance of segmentation algorithms. This would help the expert specifying high level specification of QoS, more adapted to the context, and thus improve the selection of algorithms. For instance, if an algorithm cannot cope with certain kinds of inputs, then it should be well-documented, so that its *robustness* can be known before invoking the algorithm. In [17], the authors consider precision (reliability), accuracy (validity), and efficiency (viability) and describe a framework for evaluating image segmentation algorithms.

These research works are closely akin to common examples of validation criteria proposed in [11]. Finally, it should be noted that precision, accuracy, and efficiency factors have a complex interdependency: an attempt to increase accuracy may imply a decrease in efficiency and/or precision [15].

2.3 Impact of Variability for Grid Services

Managing QoS on the grid is a crucial issue as providing end-to-end qualities is one of the topmost user requirements [10]. QoS issues have not been addressed very well in most Grid workflow management systems while supporting QoS at both specification and execution level becomes increasingly critical [19][4]. In addition to the QoS related to the grid infrastructure, attention must be put on the QoS offered by the services themselves. QoS-aware workflow engines are able to ensure that each application meets its user requirements. To do so, five relevant QoS dimensions are generally considered: time, cost, fidelity, reliability and security [19][6]. These are high level concepts that should be refined into fine-grained QoS characteristics. Moreover, QoS grid concerns such as reliability, latency or security can be expressed through this classification. The analysis of variability in medical imaging has shown that such dimensions are also considered in the domain [11], and especially the fidelity dimension.

The QoS variability of medical images segmentation services impacts various operations in the management of workflows. For instance, for a given task of the workflow, the most adapted service can be selected considering QoS attributes of the service and the clinician requirements. Our service product line must facilitate QoS management in relation with workflow engines, so that:

- Some resulting QoS of an application can be computed *before* making the service available to customers. Application of statistical methods are likely to be conducted in this case.
- The selection (statically or dynamically) of services through their QoS can be made by the workflow engine. Scheduling of computational tasks on the grid is a complex optimization problem which may require different criteria to be considered [18]. Few research works propose a scheduling approach for multiple criteria on QoS parameters, and they mainly focus on QoS aspects of the grid infrastructure itself [3]. Our aim is thus to manage QoS on medical imaging characteristics and on properties of the grid at the same time. Just as segmentation algorithms can be provided with variable QoS attributes and deployed on the grid, the service product line should enable one to use such a scheduling approach.
- The appropriate QoS monitoring is implemented. Monitoring systems can control the fulfillment of QoS criteria and moreover offer error detection and recovery. For instance, discrepancy methods can dynamically check the process results and an erroneous segmentation process should be detected early. Consequently, the service product line should handle fine enough information on QoS so that monitoring constraints can be described and reused.
- Adaptation strategies can be implemented as well. In response to unexpected behaviour – detected by the above monitoring system – or technical conditions (delay, latency), it is necessary to adapt or reschedule a workflow, considering a set of alternative services. The service product line should be also usable in this context.

3 Building the Service Product Line

In this section we first present the concept of software product lines, which objective is to handle the identified variability points. We specify the architecture of intended *service product line* and discuss open issues regarding its implementation and integration.

3.1 Software Product Lines

Manufacturers have long employed analogous engineering techniques to create a product line of similar products using a common factory that assembles and configures parts designed to be reused across the product line. For example, automotive manufacturers can create tens of thousands of unique variations of one car model using a single pool of carefully designed parts and one factory specifically designed to configure and assemble those parts. Similarly, software product lines (SPL) refers to engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production. A SPL must support the concepts of variability in order to describe not only one specific entity but a family of entities. This means to be able to choose en entity between several possible variants and to select optional parts. This process is called the derivation process. If we compare it to the concept of generic class in object-oriented languages, the description of one family of entities corresponds to the description of a generic class and the derivation process corresponds to the instantiation of the generic class. Like the instantiation process needs to check for the type compatibility of generic parameters, the derivation process needs to check that the constraints that exist between several optional parts or variants (either mutual exclusion or dependancy) are verified by the user choices. At the code level, conditional compiling and aspect-oriented programming have been studied to manage variability at implementation and compile time [1]. More recently, variability management has gained attention in the earlier steps of the software development [21]. In [14] we have proposed to introduce variability into an

Aspect-Oriented Modeling (AOM) approach and use it to design SPL. Besides this approach is now getting more attention in the SOA domain [7].

Therefore, we want to provide to software architects capabilities to manipulate an imaging service not only as one service but as a service product line, which allows them to build easily derived services that include the functionalities and the QoS properties matching their requirements. Considering the mentioned requirements, our purpose is to tackle the determined variability issues by including services within a product line architecture. In our context, Model-driven engineering (MDE) techniques are intended to be used to capture the description of service variability and product-line capabilities. MDE constitutes the most recent evolution of models usage in software engineering [12]. MDE generalizes the usage of models and put them at the core of the software development process. Moreover, MDE is a promising approach to address platform complexity and to express domain concepts through domain-specific modeling languages described using metamodels. Models transformation ensure the consistency between application implementations and analysis information associated with functional and QoS requirements captured by models (*i.e* transformation of platform independent models into platform-specific models). Finally, we will use those models in order to generate the intended SPL behaviour on various grid infrastructures.

3.2 Principles

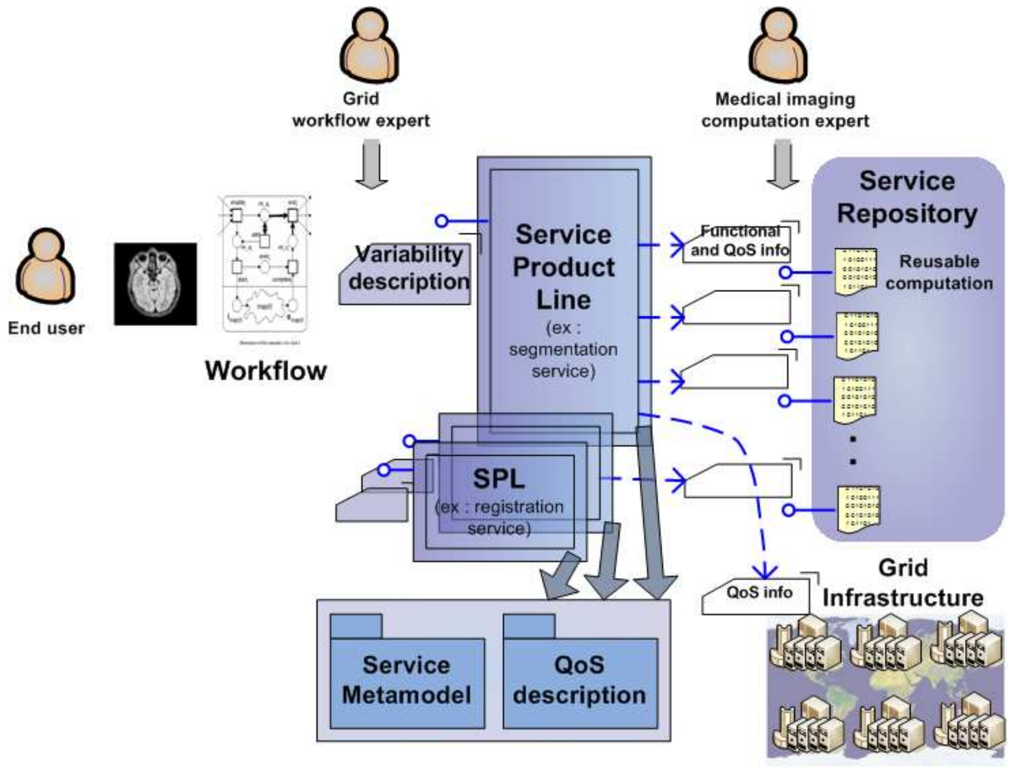


Figure 1: Overview of the Software Product Line Framework

Figure 1 sets the principles of our approach. It relies on i) a service product line framework (SPLF) describing the business domain, ii) a service repository, containing legacy services of the business domain and iii) metamodels, which capture the knowledge of the SPLF. Some metamodels specify SOA and QoS information for handling variability and other ones describe valuable information from grid infrastructures, which have an impact on SPL variability.

The software product line framework (SPLF). It describes possible workflows for the domain of medical imaging. It considers two levels of variability, *i.e* workflow variability and service variability. At both levels there are:

- a set of common properties (a structured list of assumptions that are true for all members of the domain). For example, any service for image segmentation requires a medical image as input;
- a set of possible differences. For example the format of medical image may vary depending on the service that is chosen (DICOM, Nifti, Analyze, *etc.*), so that it is necessary to record that a service supports or not a given format. This is an example of a variation point identified at the service level. The choice to insert or not one type of service (for example registration or segmentation) is another variation point, but at the workflow level.

The service repository. Services of the repository are making a collection of algorithms for image processing. According to the SPLF description, one or several services may participate to one of its possible workflows. In this repository one can find for example a subset of services which are dedicated to image segmentation. Intuitively these services may be handled through an actual service (a service interface) of type *Segmentation*, which is included in one SPL description. It makes possible to consider commonalities and to set the properties corresponding to the variation points of this type of service. In other words, one service of the repository can be considered as the result of a derivation process of a dedicated service product line. Consequently, a line of services can be seen as a service that is able to provide access to multiple services, members of the line. The result is the construction of a generic interface, which describes indirectly multiple interfaces [9].

Metamodels. Variability of grid services for medical image analysis is captured in a metamodel. It makes possible to reason on services and to achieve the operations already mentioned in section 2, such as selection, adaptation and monitoring. The SPLF relies on this metamodel to represent a software product line, which corresponds to one of its instances. As the metamodel could describe all possible software product lines of the business domain (medical imaging), it is able to represent all functional and non-functional commonalities and variations of the services belonging to the repository. Each service of this repository, which belongs to the software product line, must conform to a given model and by extension (transitivity) to the metamodel. Thus, the software architect can infer a software product line considering some services of the repository. Our metamodel helps to structure the necessary information associated to services. Semantics and knowledge are used to enhance Grid functionalities: ontology-based semantic modeling is used to enhance service-based programming on the Semantic Grid. Besides, in our preliminary implementation, we rely on an ontological approach already developed and which provides medical imaging knowledge [16]. We use feature models technology and part of our metamodels can be seen as a view on this ontology [8]. For instance, medical imaging computation experts will be able to express that a segmentation algorithm has been designed to treat brain MRIs images in DICOM format and in a precise acquisition context (*e.g* acquisition equipment).

To handle the strong needs on QoS variability, a subset of the metamodel is dedicated to QoS. For instance such information is intended to be used to compare the QoS of the members of a software product line. In section 2, the analysis of evaluation mechanisms for segmentation algorithms exhibits an important variability. The variability of QoS processing mechanisms also impacts the operations to select services, control or adapt the workflow. Selecting a service according to QoS constraints implies that the service can evaluate *a priori* the QoS dimensions. On the contrary, a few services are only able to support dynamic computation and requires the knowledge of the output; and in this case, these services are well-adapted to perform appropriate monitoring at runtime. That is why our metamodels expresses also variability in QoS mechanisms. Besides, the impact of the grid on the SPL is intended to be handled through another metamodel that records information of the grid infrastructures.

Product derivation process. Thanks to the SPLF, the repository and the metamodels, it is possible to derive services from a given software product line. The main focus during product derivation is on satisfying complex dependencies, *i.e* dependencies that affect the binding of a large number of variation points, such as quality attributes. A key aspect in resolving these dependencies is to have an overview on these complex dependencies and how they mutually relate. An example of a complex dependency is a restriction on memory usage of a software system. An example of a relation to other dependencies is how this restriction interacts with a requirement on the performance. These examples show the need for the first-class representation of dependencies, including complex dependencies, in variability models and the need for appropriate means to model the relations between these dependencies. Thus, the SPL provides to software architect a generic interface describing the set of functional and non functional characteristics and means to express constraints in order to choose the most adapted service for each workflow task.

3.3 Open Issues

According to the principles described in Section 3.2, the service product line framework is intended to provide several functionalities, but their realization makes some open issues arise. We plan to tackle all these issues by the provision of specific operations on the service and QoS models used by the SPLF. We summarize these issues as the capability:

- to specify non functional properties toward different perspectives, according to multiple levels of abstraction, and used by all actors of the grid. Moreover, different views of QoS may cooperate. In particular, the proposed framework should allow one to transform high-level QoS properties, described by the praticitioner and guided by the clinical context, into fine-grained non functional properties of medical imaging services. For this purpose, model transformation – a key notion in the MDE approach – will be used.
- to provide to the workflow middleware means to select the best services of the repository, from the software product line, considering specific QoS properties. One possibility is that workflow engines ask to the software product line qualities offered by its members. The workflow scheduler then gets information elements in order to reason on abilities of members of the software product line. Another option is that the reasoning process is directly delegated to the software product line, which would be able to instantiate the member of the line most adapted to the associated context and constraints.
- to infer a software product line considering some of the services of the repository, by detecting automatically their common elements and their variation points.
- to master (statically or dynamically) in the derivation process i) the uncertainty of the behaviour of medical imaging services, for instance the subjectivity or even the impossibility to compute the QoS of services, ii) the context-dependency (*i.e* medical or grid context) of elements of services, iii) the intradependencies between the elements of the service (*i.e* intradependencies between QoS offered by services).

4 Conclusion

This paper has proposed to tackle the variability issues in grid services for medical imaging by using an approach based on software product lines. Functional and non functional variability of imaging services have been analysed using the segmentation step as a running example. As a start, we focused on issues in building a product line on services only, providing a service product line framework. This line will notably enable service providers and workflow experts to capture the commonalities and the differences of legacy services and to use the line to select appropriate services according to functional and non functional criteria. The service product line framework is currently undergoing implementation. All metamodels are currently operational and first validation on specific medical imaging workflows are going to start. Our long term goal is to provide a complete *service product line* that would describe major variations in functional and non functional medical imaging service specifications, as well as in process chains described through workflows. As the medical imaging field exacerbates the variability problems, we also expect that the resulting solutions are going to be applicable to other data-intensive uses of grid infrastructures and general service oriented architectures.

References

[1] Michalis Anastasopoulos and Dirk Muthig. An evaluation of aspect-oriented programming as a product line implementation technology. In *ICSR*, volume 3107 of *Lecture Notes in Computer Science*, pages 141–156. Springer, 2004. 3.1

[2] Jan Bosch, Gert Florijn, Danny Greefhorst, Juha Kuusela, J. Henk Obbink, and Klaus Pohl. Variability issues in software product lines. In Frank van der Linden, editor, *PFE*, volume 2290 of *Lecture Notes in Computer Science*, pages 13–21. Springer, 2001. 1

[3] Ivona Brandic, Siegfried Benkner, Gerhard Engelbrecht, and Rainer Schmidt. Qos support for time-critical grid workflow applications. *e-science*, 0:108–115, 2005. 2.3

[4] Ivona Brandic, Rainer Schmidt, Gerhard Engelbrecht, and Siegfried Benkner. Towards quality of service support for grid workflows. In *Proceedings of the European Grid Conference 2005 (EGC2005)*, Amsterdam, The Netherlands, 2 2005. Springer Verlag. 2.3

[5] J. S. Cardoso and L. Corte-Real. Toward a generic evaluation of image segmentation. *Image Processing, IEEE Transactions on*, 14(11):1773–1782, 2005. 2.1

[6] Jorge Cardoso, Amit P. Sheth, John A. Miller, Jonathan Arnold, and Krys Kochut. Quality of service for workflows and web service processes. *J. Web Sem.*, 1(3):281–308, 2004. 2.3

[7] Sholom Cohen, editor. *Service Oriented Architectures and Product Lines - What is the Connection? SOAPL'07, Workshop at SPLC'07, 10 September 2007, Kyoto*, 2007. 1, 3.1

[8] Krzysztof Czarnecki, Chang Hwan Peter Kim, and Karl Trygve Kalleberg. Feature models are views on ontologies. In *SPLC '06: Proceedings of the 10th International on Software Product Line Conference*, pages 41–51, Washington, DC, USA, 2006. IEEE Computer Society. 3.2

[9] Cristina Feier, Dumitru Roman, Axel Polleres, John Domingue, Michael Stollberg, and Dieter Fensel. Towards intelligent web services: Web service modeling ontology (wsmo). In *Proceedings of the International Conference on Intelligent Computing (ICIC)*, Hefei, China, 8 2005. 3.2

[10] Ian Foster. What is the grid? - a three point checklist. *GRIDtoday*, 1(6), July 2002. 2.3

[11] P. Jannin, J. Fitzpatrick, D. Hawkes, X. Pennec, R. Shahidi, and M. Vannier. Validation of medical image processing in image-guided therapy, 2002. 2.2, 2.3

[12] Stuart Kent. Model driven engineering. In Michael J. Butler, Luigia Petre, and Kaisa Sere, editors, *IFM*, volume 2335 of *Lecture Notes in Computer Science*, pages 286–298. Springer, 2002. 3.1

[13] Charles W. Krueger. Towards a taxonomy for software product lines. In Frank van der Linden, editor, *PFE*, volume 3014 of *Lecture Notes in Computer Science*, pages 323–331. Springer, 2003. 1

[14] Philippe Lahire, Brice Morin, Gilles Vanwormhoudt, Alban Gaignard, Olivier Barais, and Jean-Marc Jézéquel. Introducing Variability into Aspect-Oriented Modeling Approaches. In *ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems (Models 2007)*, volume LNCS 4735 of *LNCS*, pages 498–513, Nashville TN USA, October 2007. Vanderbilt University, springer-verlag. 3.1

[15] Aleksandra Popovic, Matas de la Fuente, Martin Engelhardt, and Klaus Radermacher. Statistical validation metric for accuracy assessment in medical image segmentation. *International Journal of Computer Assisted Radiology and Surgery*, 2(3-4):169–181, December 2007. 2.2

[16] Lynda Temal, Michel Dojat, Gilles Kassel, and Bernard Gibaud. Towards an ontology for sharing medical images and regions of interest in neuroimaging. *Journal of Biomedical Informatics*, 2008. To appear. 3.2

[17] Jayaram K. Udupa, Vicki R. Leblanc, Ying Zhuge, Celina Imielinska, Hilary Schmidt, Leanne M. Currie, Bruce E. Hirsch, and James Woodburn. A framework for evaluating image segmentation algorithms. *Computerized Medical Imaging and Graphics*, 30(2):75–87, March 2006. 2.2

[18] Marek Wieczorek, Andreas Hoheisel, and Radu Prodan. Taxonomy of the multi-criteria grid workflow scheduling problem. In *CoreGrid Workshop*, 2007. 2.3

[19] J. Yu and R. Buyya. A taxonomy of workflow management systems for grid computing, 2005. 2.3

[20] Yu Jin Zhang. A review of recent evaluation methods for image segmentation. In *Signal Processing and its Applications, Sixth International, Symposium on. 2001*, volume 1, pages 148–151, Kuala Lumpur, Malaysia, 2001. 2.1, 2.2

[21] Tewfik Ziadi and Jean-Marc Jézéquel. *Software Product Lines*, chapter Product Line Engineering with the UML: Deriving Products, pages 557–586. Number ISBN: 978-3-540-33252-7. Springer Verlag, 2006. 3.1