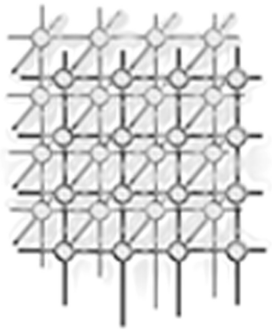
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2009; **00**:1–7



A Grid Workflow Environment for Brain Imaging Analysis on Distributed Systems

Suraj Pandey*,†[1], William Voorsluys[1] , Mustafizur Rahman[1], Rajkumar Buyya[1], James Dobson[2], Kenneth Chiu[3]

[1] *Grid Computing and Distributed Systems (GRIDS) Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia*
[2] *Department of Psychological & Brain Sciences, Dartmouth College, USA*
[3] *Grid Computing Research Lab, Department of Computer Science, State University of New York (SUNY) at Binghamton, NY, USA*

SUMMARY

Scientific applications like neuroscience data analysis are usually compute and data-intensive. With the use of the additional capacity offered by distributed resources and suitable middlewares, we can achieve much shorter execution time, distribute compute and storage load, and add greater flexibility to the execution of these scientific applications than we could ever achieve in a single compute resource.

In this paper, we present the processing of Image Registration (IR) for Functional Magnetic Resonance Imaging (fMRI) studies on Global Grids. We characterize the application, list its requirements and then transform it to a workflow. We use Gridbus Broker and Gridbus Workflow Engine (GWFE) technologies for executing the neuroscience application on the Grid. We developed a complete web-based portal integrating GUI-based workflow editor, execution management, monitoring and visualization of tasks and resources. We describe each component of the system in detail. We then execute the application on Grid'5000 platform and present extensive performance results. We show that the IR application can have 1) significantly improved makespan, 2) distribution of compute and storage load among resources used, and 3) flexibility when executing multiple times on Grid resources.

KEY WORDS: Brain Imaging, Workflow, Grid Computing, Distributed Systems

*Correspondence to: S. Pandey, 5.30a, ICT Building, 111 Barry St., Carlton 3053, Victoria, Australia.
†E-mail: spandey@csse.unimelb.edu.au

Received 6 November 2008
Revised 6 November 2008

1. INTRODUCTION

Nowadays many scientific experiments such as climate modeling, structural biology and chemistry, neuroscience data analysis, and disaster recovery are conducted through complex and distributed scientific computations that are represented and structured as scientific workflows [13]. Representing these application in the form of a workflow highly simplifies the layout of individual or a group of components of the application as compared to the raw form (usually scripts). The components of a workflow are usually tasks and data linked together using dependency rules and control sequences. Scientific workflows usually need to process a huge amount of data and are computationally intensive activities. Neuroscience data analysis is one such application that has been a focus of much research in recent years (NIFTI, BIRN) (see Section 2).

The neuroscience data analysis application we present in this paper has several tasks that need to be structured according to their data dependencies for correct execution. Both the data and computation requirements are very high, depending on the number of subjects analyzed. Given the typically large number of subjects' data being analyzed, it takes significant amount of time for this application to produce results when executed as a sequential process on limited resources. Moreover, scientists may need to re-run the application by varying run-time parameters. Often researchers and users around the globe may share the results produced. To facilitate these requirements such as high compute power, repeated experiments, sharing of data and results, this application may leverage the power of distributed resources presented by platforms such as Grids. By executing this application on distributed resources execution time can be minimized, repeated executions can be performed with little overhead, reliability of execution can be increased, and resource usage can be distributed. It is a very demanding task for researchers to handle these complex applications directly on Global Grids without proper management systems, interfaces, and utilities. Therefore, user friendly systems are increasingly being developed to enable e-scientists to integrate, structure and orchestrate various local or remote data and service resources to perform scientific experiments to produce interesting scientific discoveries.

A scientific workflow management system is one of the popular approaches that provide an environment for managing scientific experiments, which have data dependent tasks, by hiding the orchestration and integration details inherent while executing workflows on distributed resources.

The Gridbus Workflow Engine (GWFE) [23], is one such Grid-based workflow management system that aids users (scientists) by enabling their applications to be represented as a workflow and then execute on the Grid from a higher level of abstraction. The GWFE provides an easy-to-use workflow editor for application composition, an XML-based workflow language for structured representation, and a user-friendly portal with discovery, monitoring, and scheduling components that enables users to select resources, upload files and keep track of the application's progress.

In this paper, we present a Brain Imaging application that performs Image Registration (IR) which can be used for Functional Magnetic Resonance Imaging (fMRI) studies. We first characterize the application and identify its requirements. We describe the components of GWFE that enable the application to leverage the capabilities of Grid. We then execute

this application on Global Grids and present detailed analysis of the results. Our performance results and technology used could directly assist neuroscientists using brain imaging technology in clinical areas such as epilepsy, stroke, brain trauma and mental health.

Our main contributions in this paper are as follows:

1. representation of a brain imaging application in the form of a workflow.
2. design and implementation of a tool set to manage a complete life cycle of the workflow.
3. characterization of the tasks of brain imaging application workflow in terms of data and computational requirements.
4. a performance evaluation of the workflow execution on the Grid.

The rest of the paper is structured as follows: Section 2 presents related work and Section 3 describes the workflow application scenario, its components, and requirements. In Section 4, we describe the workflow execution life cycle and the major components that users can use to interact with the system. Section 5 presents an experimental evaluation of a real biomedical application by executing it on the Grid. Section 6 concludes the paper and discusses some future work.

2. RELATED WORK

Several projects are investigating workflow technology with respect to our target application and Global Grid scheduling. These related works are categorized as **application** and **Workflow Management Systems** and described below.

Application: Olbarriaga et al. [15] present the Virtual Laboratory for fMRI (VL-fMRI) project, whose goal is to provide an IT infrastructure to facilitate management, analysis, and sharing of data in neuroimaging research with a focus on functional MRI. We share a common objective to facilitate the data logistics and management in fMRI analysis via workflow automation. Their system could use our workflow management system as a pluggable component.

Neurobase [10] uses grid technology for the integration and sharing of heterogeneous sources of information in neuroimaging from both data and computing aspects.

Buyya et al. [3] studied instrumentation and distribution analysis of brain activity data on global grids. They present the design and development of Magnetoencephalography (MEG) data analysis system. They describe the composition of the neuroscience application as parameter-sweep application and its on-demand deployment on Global Grids.

Ellis et al. [9] executed their IR algorithm by registering several couples of T1 MRI images coming from different subjects in 5 minutes on a Grid consisting of 15 2GHz Pentium IV PCs linked through a 1Gigabit/s network. This is an example where the capabilities of Grid has been used to speedup brain imaging applications.

The LONI Pipeline [17] was developed to facilitate ease of workflow construction, validation and execution like many similar workflow environments, primarily used in the context of neuroimaging. This initiative, which was as early as 2003, clearly demonstrates that workflow technology can be used and is viable for neuroimaging applications.

The NIMH Neuroimaging Informatics Technology Initiative (NIFTI[†]) was formed to aid in the development and enhancement of informatics tools for neuroimaging. Likewise, the Biomedical Informatics Research Network (BIRN[‡]) is another high profile effort working to develop standards (eg. LONI) among its consortia membership. Such efforts are contributing more towards standards, efficiency, interoperability and integration of tools. We have tried to use these tools to harness the power of distributed resources to increase several characteristics of the application.

Workflow Management Systems: Deelman et al. [7] have done considerable work on planning, mapping and data-reuse in the area of workflow scheduling. They propose Pegasus [8], which is a framework that maps complex scientific workflows onto distributed resources such as the Grid. DAGMan, together with Pegasus, schedules tasks to Condor system. With the integration of Chimera [11] and Pegasus based [7] mapping, it can execute complex workflows based on pre-planning. In our system, we use dynamic mapping of tasks to resources based on current resource availability.

The Taverna project [14] has developed a tool for the composition and enactment of bioinformatics workflows for the life science community. This tool provides a graphical user interface for the composition of workflows. Other well-known projects on workflow systems include GridFlow [4], Unicore [18], ICENI [12], GridAnt [2] and Triana [20].

Yu et al. [22] proposed a comprehensive taxonomy on workflow management systems. Chen et al. [6] proposed a taxonomy for workflow verification and validation. We refer the reader to these taxonomies for the characterization and classification of existing workflow management systems.

3. A SCENARIO AND REQUIREMENTS

Image registration is a brain imaging technique. We describe the Image Registration (IR) procedure as a Scientific Workflow Application. We construct the IR workflow and describe each of its tasks, and then tabulate the requirements of executing each task in the workflow.

fMRI and IR: fMRI attempts to determine which parts of the brain are active in response to some given stimulus. For instance, a person (referred as subject in this paper), in the Magnetic Resonance (MR) scanner, would be asked to perform some tasks, e.g., finger-tap at regular intervals. As the subject performs the task, researchers effectively take 3-D MR images of his brain. The goal is to identify those parts of the brain responsible for processing the information the stimulus provides.

IR is ubiquitous in fMRI analysis, especially in the case of multi-subject studies. IR is the process of estimating an optimal transformation between two images, also known as "Spatial

[†]http://nifti.nimh.nih.gov
[‡]http://nbhirn.net

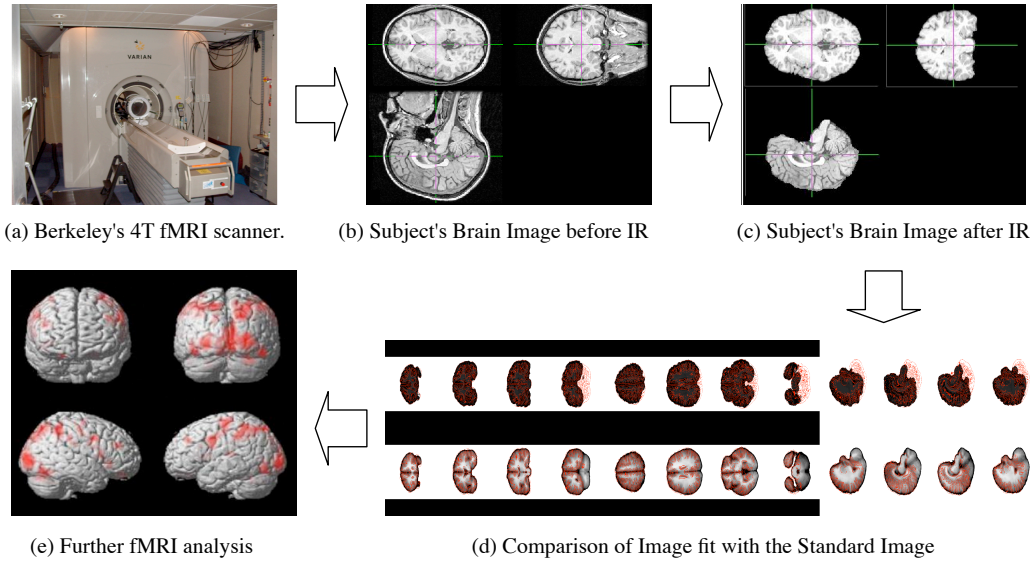


Figure 1. Image Registration and fMRI.

Normalization" in functional neuroimaging [16]. When registering images we are determining a geometric transformation, which aligns one image to fit another. The aim is to establish a one-to-one continuous correspondence between the brain images of different individuals. The transformation will reduce the anatomical variability between high-resolution anatomical brain images from different subjects. This enables analysts to compute a single activation map representing the entire group of subjects or to compare the brain activation between two different groups of subjects.

The IR procedure and its relation to fMRI is depicted in Figure 1. The scanner acquires high-resolution images of each subject's brain. Due to subject movements, the images can be oriented in different positions at the time of scanning. One such image of a subject before registration is shown in Figure 1 (b). The registration process ensures that all the images of different subjects are normalized against a standard image and in a common 3D space. The normalized image of the subject is shown in Figure 1 (c). After normalization, the subject's normalized image is compared with the *atlas* (reference image) for the quality of fit. This comparison is shown in Figure 1 (d). The workflow we study in this paper, first produces the *atlas*, then produces the comparison image (Figure 1 (d)) as output for each subject.

Application Description: The IR procedure, expressed as a scientific workflow is shown in Figure 2. The tasks are linked according to their data dependencies. Individual tasks that form the workflow are described below [19] [1].

BET: (Brain Extraction Tool) deletes non-brain tissue from an image of the whole head and extracts brain's image.

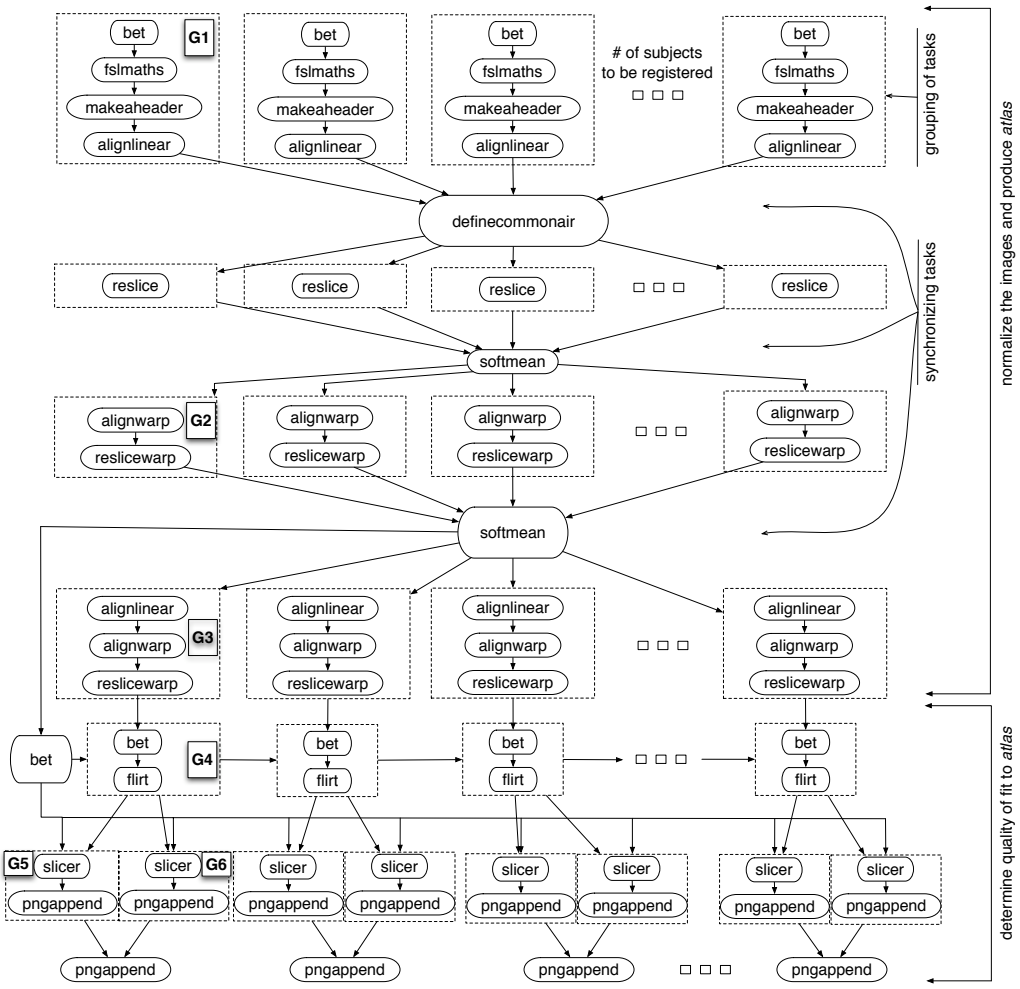


Figure 2. Image Registration Workflow.

FSLMATHS: allows mathematical manipulation of images.

MAKEAHEADER: generates a header (.hdr) file based on the parameters (type, x-y-z dimensions and size).

ALIGNLINEAR: is a general linear intra modality registration tool. Any image can be aligned to a representative with a transformation model using alignlinear. It generates *.air* files that can be used to reslice the specified reslice data set to match the specified standard data set. We use affine 12-parameter full-afine model.

DEFINECOMMONAIR: defines new *.air* files with a new standard file that defines the "average" position, size and shape of the various reslice files.

RESLICE: takes a *.air* file and uses the information that it contains to load the corresponding image file and generate a new, realigned file.

SOFTMEAN: averages together a series of files.

ALIGN_WARP: compares the reference image to determine how the new image should be warped, i.e. the position and shape of the image adjusted, to match the reference image. It is a nonlinear registration tool that generates a *.warp* file that can be used to reslice the specified reslice data set to match the specified standard data set.

RESLICE_WARP: takes a *.warp* file and uses the information that it contains to load the corresponding image file and generate a new, realigned file.

FLIRT: performs affine registration. It produces an output volume, where the transform is applied to the input volume to align it with the reference volume (*atlas* created in previous steps).

SLICER: takes 3D image and produces 2D pictures of slices.

PNGAPPEND: processes addition/subtraction of *.png* images.

Application Requirements: According to Zhao et.al [24], in a typical year the Dartmouth Brain Imaging Center about 60 researchers pre-process and analyze data from about 20 concurrent studies. The raw fMRI data for a typical study would consist of three subject groups with 20 subjects per group, five experimental runs per subject, and 300 volume images per run, yielding 90,000 volumes and over 60 GB of data. Intensive analysis begins once the images are processed. IR forms a part of the image pre-processing step using only the high-resolution data, which represents a minor portion of the entire workflow's execution time.

We characterize each task in the IR workflow in Table I. It lists each task, its input files and sizes, its average computation time ($\bar{w_i}$) on a single machine, and standard deviation (σ) computed over 40 subjects on a set of 10 random machines in Grid'5000 [5]. The random machines chosen didn't vary much in their processing powers. The high values of standard deviation (σ) for certain tasks can be explained by examining the nature of the operation of that task. In this application, tasks having longer execution time have higher values of deviation. The execution time also depends on the orientation of the image to be aligned.

A complete execution of the workflow of 40 subjects on a single CPU with single core (without local load) takes two and a half days to complete. The total storage space needed for the complete execution of 40 subjects exceeds 20GB on a single machine when the intermediate files are retained. Moreover, the computation time and storage requirements limit the number of subjects that can be used for execution at one time on a single machine.

When the application is executed on distributed resources with no resource scarcity the application should take as much time to execute all the subjects as a single machine would take to execute a single subject workflow without local load. However, the real execution time is higher than the ideal case (∼69 minutes) for 1 subject as shown in Table I, due to the significant amount of time taken to transfer the intermediate files from one resource to another. We can decrease the transfer time by allowing multiple tasks to run on a single machine (grouping of tasks). Also, the synchronizing tasks take longer to execute when there are more subjects. The

Table I. Characteristics of tasks for a single subject's IR.
note: X = hires, '–' = not applicable (depends on # of subjects),
taskname(*) = same task but different execution instance, N = subject index ($N \in \mathbb{Z}$)

#	Task Name	Input Files	Source Tasks	Size of i/p (MB)	$\bar{w}_i$(sec)	σ
1	bet(1)	X.{hdr,img}	Staging server	16	45	11.91
2	fslmaths	bX.{hdr,img}	bet(1)	16	1	0.42
3	makeaheader	bX.{hdr,img}	fslmaths	16	$\ll 1$	–
4	alignlinear(1)	bX.{hdr,img}	fslmaths	16	2	0.47
5	definecommonair	X.air, bX.{hdr,img}	alignlinear(1)	16	94	–
6	reslice	X.air.aircommon, bX.{hdr,img}	definecommonair	16	5	0.5
7	softmean(1)	X-reslice.{hdr,img}	reslice	20	140	–
8	alignwarp(1)	atlas-linear.{hdr,img}, X-reslice.{hdr,img}	softmean(1)	40	971	620.17
9	reslicewarp(1)	atlas-linear.{hdr,img}, X-reslice.{hdr,img,warp}	alignwarp(1)	40	9	1.88
10	softmean(2)	X-reslice-warp.{hdr,img}	reslicewarp(1)	20	111	–
11	bet(2)	atlas.{hdr,img}	softmean(2)	20	11	1.5
12	alignlinear(2)	bX.{hdr,img}, atlas.{hdr,img}	definecommonair, softmean(2)	36	23	10.25
13	alignwarp(2)	X.air, atlas.{hdr,img}, bX.{hdr,img}	alignlinear(2)	36	2656	1501
14	reslicewarp(2)	bX.{hdr,img,warp}	alignwarp(2)	16	9	1.88
15	bet(3)	nX.{hdr,img}	reslicewarp(2)	16	15	1.3
16	flirt	batlas.{hdr,img}, nbX.{hdr,img}	bet(2), bet(3)	56	6	0.44
17	slicer(1)	batlas.{hdr,img}, N-fit.{hdr,img}	bet(2), flirt	80	8	0.44
18	pngappend(1)	{sla,slb,...,slk,sll}.png	slicer(1)	0.3	4	0.51
19	slicer(2)	batlas.{hdr,img}, N-fit.{hdr,img}	bet(2), flirt	80	8	0.44
20	pngappend(2)	{sla,slb,...,slk,sll}.png	slicer(2)	0.3	4	0.28
21	pngappend(3)	{N-fit1, N-fit2}.png	pngappend(1), pngappend(2)	0.8	4	0.28
22	OUTPUT	**N-fit.png**	pngappend(3)	(o/p size) 0.8		
	Average data volume and computation time:			**558.2 MB**	**∼69min**	

coordination time taken by the middleware also adds to the overall increase in total execution time.

Application to End-users and Challenges: In an effort to improve the quality of image registration and to provide clean, high-resolution images for 3D display, researchers have been collecting multiple T1-weighted structural MRI volumes. Recent projects have used up to four of these volumes per subject. The resulting average volume can then be combined with the larger subject population in order to produce the probabilistic atlas. In additional to these anatomically derived processes researchers have begun experimenting with the use of methods that will use functional information to register data collected in a time-series. The introduction

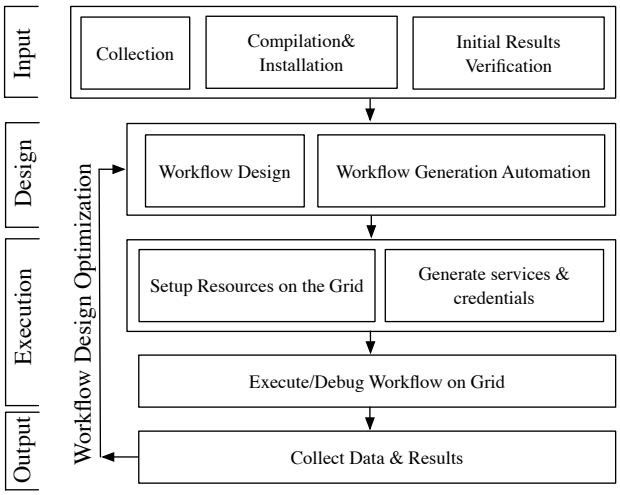


Figure 3. Deployment Cycle.

of new protocols and the acquisition of multiple high-resolution volumes have increased both the time to acquire and pre-process a typical study. In order to facilitate these new methods imaging centers will need to provide additional data storage and compute capacity. These centers will most likely need to create a shared database of subjects, along with the increased variety of imaging modalities collected, and to expose to individual investigators the methods used to calculate average structural volumes.

Typical fMRI experiments have between twenty and thirty subjects with some having over fifty subjects. A number are removed from the analysis, often due to excessive head movement that image registration algorithms are unable to correct for. While the image registration application described in this paper makes use of only a single high-resolution volume per subject, the addition of several more volumes would be trivial. Doing such would enable both a cleaner volume for the subject as well as a tighter fit to the atlas. Processes such as these are often repeated many times with spot checks at critical points, such as during the first average. In the case of a poor fit to the atlas, or an outlier distorting the overall registration, modifications can be made to the workflow. These modifications might include to the rejection of a subject or a change in application parameters.

4. WORKFLOW MANAGEMENT ON THE GRID

We describe a process of constructing and experimenting with the workflow on the Grid. We then present the components of Gridbus Workflow Management System (GWMS).

4.1. Workflow Deployment Cycle

The life cycle of deployment of the workflow is depicted in Figure 3. In the *input* phase, scientists provide the input data, batch scripts, sample output files, and application requirements. In order to run the application on the Grid with heterogeneous resources, the executables are required to be compiled and installed (can be submitted at runtime) at both remote and local sites. For quality assurance or *initial results verification*, this step involves the testing of conformance of our execution with that of the sample results provided by the scientists. Once the initial results are verified the workflow structure and its details need to be composed in the *designing* phase. In this phase, the automated generation of the workflow in terms of the workflow language used can also be done by taking into account the run-time parameters that users might want to change during execution.

In the *Execution* phase the resources, where the application can be executed, need to be setup. The resource list, its credentials and the services provided by each resource need to be inserted into the catalogue. When experiments are conducted repeatedly and in time, resource availability and conditions will have changed. This requires services and credentials list to be generated for each execution with the help of the catalogue. The application is then executed on the Grid. Usually debugging and testing is done while the application is being executed, but this depends on the software development process being used. Depending on the analysis of the results from the *output* phase, the workflow design can be further optimized according to the requirements of the user. These are the basic steps involved in constructing most of the scientific workflows to be executed on the Grid, but doesn't generalize all applications. The process can get complicated when more user and system requirements are added.

4.2. Components of GWMS

Users interact with the GWMS through the Grid Portal. Figure 4 depicts the components of GWMS. We describe the following key components that provide users access to the scientific application.

Grid Portal: The primary user interface for our IR application is a Web Portal that encompasses the following functionalities:

1. A workflow editor, which enables users to compose new workflows and modify existing ones.
2. A submission page, through which users can upload to the system, all necessary input files to run a workflow including the workflow description file, credentials, and services files (Figure 8 (b)).
3. A monitoring and output visualization page, which allows users to monitor multiple workflow executions in progress. The most common monitoring activity consists of keeping track the status of each task through the workflow monitor, which provides a real-time updated graphical representation of workflow tasks. The application's output is presented in the form of images (Figure 8 (d), 1 (d)).

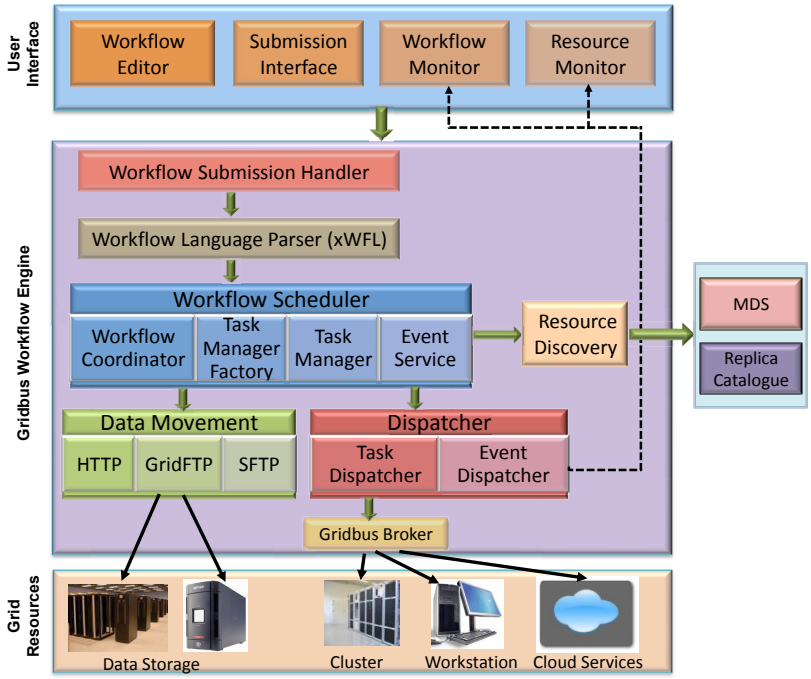
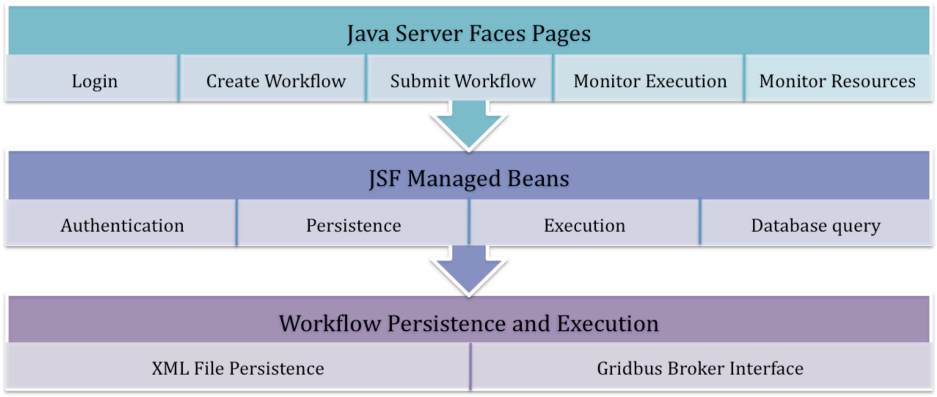


Figure 4. Components of GWMS.

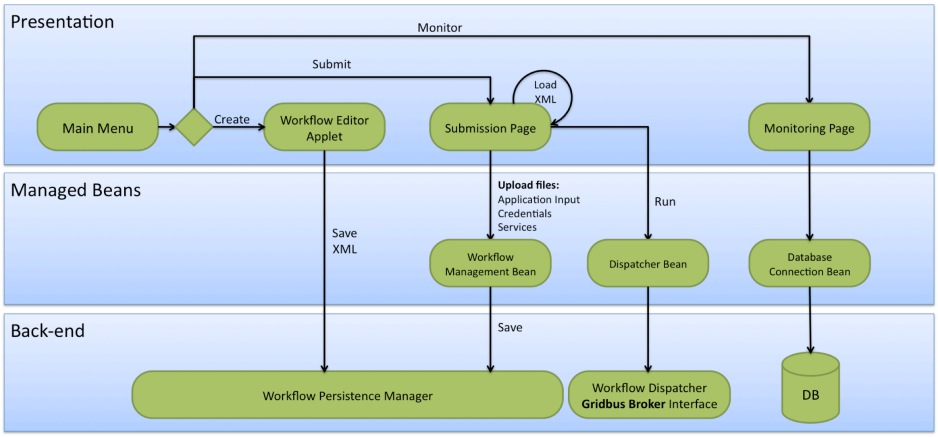
4. A resource information page, which shows the characteristics of all available grid resources.
5. An application specific page, which in the current implementation provides generation of IR workflow description files by taking the number of subjects as input.

Although the current incarnation of this Grid Portal is targeted at an IR application, our design is in no means limited to such application. Apart from few parts of the output visualization page and the application specific page, the portal infrastructure is generic enough to allow the porting of almost any workflow application into the portal.

Figure 5(a) shows a layered architecture of the GWMS portal. In the top layer, a set of Java Server Faces (JSF) pages enable actions such as creating, submitting and monitoring a Workflow execution. In the middle layer, a set of session beans manage users' requests, which in turn are forwarded to the backend (bottom) layer, which handles persistence of Workflow description and input files and their submission via the Gridbus Broker interface. Figure 5(b) depicts possible user activities and their flow through the layered portal architecture. A typical activity consists of workflow design by means of the Workflow Editor Java Applet, which generates a Workflow description XML file. Subsequently, the description file is loaded on the submission page, which requests the user to upload all input files referenced in the description file. Once all files were uploaded the submission page allows the user to submit the workflow and consequently monitor its progress.



(a)



(b)

Figure 5. Gridbus Workflow Portal. (a) Layered Architecture of the Gridbus Workflow Portal. (b) Activities the portal supports.

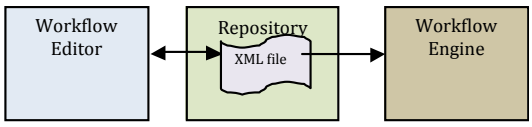


Figure 6. Interaction between Gridbus Workflow Editor and GWFE.

We now describe in more detail the Workflow Editor, the Workflow Monitor, and the Workflow Engine.

Workflow Editor: The Gridbus workflow editor provides a Graphical User Interface (GUI) and allows users to create new and modify existing workflows, based on an XML-based workflow language (xWFL) utilizing the drag and drop facilities. Using the editor, expert users can design and create the workflows for complex scientific procedures following the workflow language and the nature of application, whereas primitive users can reuse these workflows with some minor changes. In the editor, workflows are created graphically as a Directed acyclic Graph (DAG) with some nodes and links. The node represents the computational activities of a particular task in the workflow and a link is used to specify the data flow between two tasks.

Figure 7 shows the schema of task definition using xWFL. $< task >$ is a set of instructions to be executed in a resource. The element $< executable >$ is used to define the information about the task, corresponding I/O model as well as the input and output data. xWFL supports both abstract and concrete workflows. The users can either specify the location of a particular service providing a required application in $< service >$ or leave it to the engine to identify their providers dynamically at run-time. $< input >$ and $< output >$ are used to define the input and output data of a task respectively. Each data is associated with a unique port number of that task. Input can be both file and message whereas output can be only file. Figure 6 shows the schema of the links using xWFL. The element $< links >$ is a set of definitions of the links in a workflow. Each link, $< link >$ is associated with an output task (from where a file is generated) and an input task (which requires that file) as well as their corresponding port numbers.

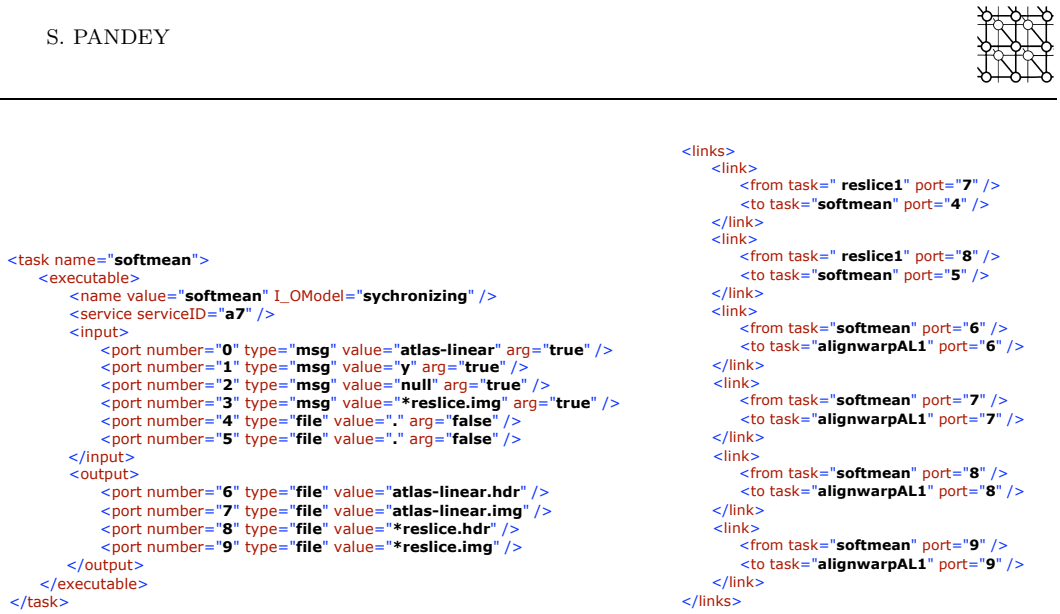
The Gridbus workflow editor provides the following advantages to the users:

Portal-based editor: As the workflow editor can be accessed through the workflow portal, users can create or edit and save their workflows in the server and can access them from anywhere and whenever required. Moreover, the users have no application dependencies on their side regarding the editor tool, and can get a pervasive access mechanism to the editor.

Hiding complexity: In general, the underlying structure of workflows is represented as XML description, which is parsed by the workflow engines. However, representing workflows in XML description demands advanced skill from workflow designers such as thorough understating of workflow language e.g. xWFL, and expertise of managing XML files from various namespaces. Gridbus workflow editor hides these complexities from scientists and researchers by providing a GUI. Users only need to draw boxes for tasks and lines for connecting them, and specify their properties. The editor compiles it and generates the XML description of the workflow automatically. Workflow editor integration with the GWFE via xWFL is shown in Figure 6.

Ease of use: Gridbus workflow editor provides a drag and drop facility to draw the workflow using boxes and lines. The common edit operations such as cut, copy, paste are provided. It also supports usual file operations such as create, open and save. Furthermore, users can simultaneously see the equivalent XML description of the graphical representation of the workflow. They can also save the graphical representation as an image.

Interoperability: The editor can generate the graphical representation of the workflow from its XML description if it is created following the schema of xWFL. Thus, if any scientist or researcher creates workflow and saves it as XML file, others can reuse that by opening it using Gridbus workflow editor, and modifying according to their needs.



```
<task name="softmean">
    <executable>
        <name value="softmean" I_OModel="sychronizing" />
        <service serviceID="a7" />
        <input>
            <port number="0" type="msg" value="atlas-linear" arg="true" />
            <port number="1" type="msg" value="y" arg="true" />
            <port number="2" type="msg" value="null" arg="true" />
            <port number="3" type="msg" value="*reslice.img" arg="true" />
            <port number="4" type="file" value="." arg="false" />
            <port number="5" type="file" value="." arg="false" />
        </input>
        <output>
            <port number="6" type="file" value="atlas-linear.hdr" />
            <port number="7" type="file" value="atlas-linear.img" />
            <port number="8" type="file" value="*reslice.hdr" />
            <port number="9" type="file" value="*reslice.img" />
        </output>
    </executable>
</task>
```

(a)

```
<links>
    <link>
        <from task=" reslice1" port="7" />
        <to task="softmean" port="4" />
    </link>
    <link>
        <from task=" reslice1" port="8" />
        <to task="softmean" port="5" />
    </link>
    <link>
        <from task="softmean" port="6" />
        <to task="alignwarpAL1" port="6" />
    </link>
    <link>
        <from task="softmean" port="7" />
        <to task="alignwarpAL1" port="7" />
    </link>
    <link>
        <from task="softmean" port="8" />
        <to task="alignwarpAL1" port="8" />
    </link>
    <link>
        <from task="softmean" port="9" />
        <to task="alignwarpAL1" port="9" />
    </link>
</links>
```

(b)

Figure 7. Defining task and links in an IR workflow using xWFL. (a) task. (b) links.

Workflow Monitor: The Gridbus Workflow Monitor provides a GUI for viewing the status of each task in the workflow. Users can easily view the ready, executing, stage-in, and completed tasks as depicted in Figure 8. Task status is represented using different colors. Users can also view the site of execution of each task, the number of tasks being executed (in case of a parameter sweep type of application) and the failure history of each task. The workflow structure is editable such that users can drag tasks and group or separate tasks of interest when there are numerous tasks in the workflow. The monitor interacts with the GWFE using an event mechanism by using the tuple space model. In the backend, a database server stores the states of each task for each application workflow. Whenever any task changes state, the monitoring interface is notified and the status values are stored. This enables multiple users to access the monitoring interface from different locations. The monitoring interface does not have support for deletion and insertion of individual tasks at run-time. However, users can add and delete tasks at the time of construction using the Gridbus editor.

Gridbus Workflow Engine: Scientific application portals submit task definitions along with their dependencies in the form of the workflow language to GWFE. Then the GWFE schedules the tasks in the workflow application through Grid middleware services and manages the execution of tasks on the Grid resources. The key components of GWFE are: workflow submission, workflow language parser, resource discovery, dispatcher, data movement and workflow scheduler.

GWFE is designed to support an XML-based WorkFlow Language (xWFL). This facilitates user level planning at the submission time. The workflow language parser converts workflow description from XML format to Tasks, Parameters, Data Constraint (workflow dependency), Conditions, etc., that can be accessed by workflow scheduler. The resource discovery component of GWFE can query Grid Information Services such as Globus MDS, directory service, and

replica catalogues, to locate suitable resources for execution of the tasks in the workflow by coordinating with middleware technologies such as Gridbus Broker [21]. GWFE uses Gridbus Broker for deploying and managing task execution on various middlewares as a dispatcher component. Gridbus Broker as a middleware mediates access to distributed resources by (a) discovering resources, (b) deploying and monitoring task execution on selected resources, (c) accessing data from local or remote data source during task execution, and (d) collating and presenting results.

GWFE is designed to be loosely-coupled and flexible using a tuple spaces model, event-driven mechanism, and subscription/notification approach in the workflow scheduler, which is managed by the workflow coordinator component. The data movement component of GWFE enables data transfers between Grid resources by using SFTP and GridFTP protocols. The workflow executor is the central component in GWFE. With the help from dispatcher component it interacts with the resource discovery component to find suitable Grid resources at run time, submits a task to resources, and controls input data transfer between task execution nodes.

Algorithm 1 Just-In-Time Scheduler

1: **for** each root task $t_i \in T_{roots}$ **do**
2: Assign t_i to an available compute resource c_k
3: **end for**
4: **repeat**
5: **for all** $t_i \in T_{nonroots}$ **do**
6: Assign ready task t_i to any available compute resource c_k
7: **end for**
8: Dispatch all the mapped tasks
9: Wait for $POLLING\_TIME$
10: Update the ready task list
11: **until** there are unscheduled tasks in the ready list

In addition to the random and round-robin scheduling policies, GWFE has a level based scheduling policy that allows the resource allocation decision to be made at the time of task submission. Algorithm 1 lists the scheduling algorithm that we used for scheduling the IR workflow. Tasks at the top level (that have no parent) are assigned to resources that have not exceeded their job limit. Tasks become ready as a result of their parents finishing execution and producing valid data. A list is formed to store these tasks. This list is updated during the *polling_time*. The scheduler then assigns these ready tasks to resources based on the availability of each resource. The optimum choice for *polling_time* depends on the number of tasks in the application, resource management policies, scheduler overhead etc.

We have a mechanism to specify the location to store the intermediate data. For data-intensive applications, such as the one presented in this paper, data cannot be stored in the compute nodes due to their size. So we store it in centralized locations. For those applications that are less data-intensive, the compute node that executes the tasks may store the data locally until the workflow finishes execution. Depending on the user's requirements, the data

can then be either migrated to a centralized server after the execution or deleted from the compute nodes after certain number of executions. Data provenance in workflows has been a challenging research topic.

In our current implementation, we handle failures by resubmitting the tasks to resources that do not have failure history for those tasks. Task resubmission and task duplication have been one of the commonly used techniques for handling failures. Fault tolerance could also be achieved by checkpointing the execution of workflows, which we leave for our future work.

5. EXPERIMENTAL EVALUATION

The IR application together with GWFE was demonstrated at the First IEEE International Scalable Computing Challenge (SCALE 2008) in conjunction with CCGrid 2008, May 19-22, 2008, using resources provided by SUNY at Binghamton and the University of Melbourne. The application was also demonstrated at the Fourth IEEE International Conference on e-Science, December 10-12, 2008. For this paper we executed the application on Grid'5000 [5]. We now describe the experiment setup, results obtained, and observations.

Table II. Grid'5000 sites used; # cores (n), # tasks (t) executed and average computation time ($\bar{c}$) (in seconds) taken on each site for each experimental group.

Site	10Sub			10Sub (G)			20Sub			20Sub (G)			40Sub			40Sub (G)		
Name	#n	#t	$\bar{c}$	#n	#t	$\bar{c}$	#n	#t	$\bar{c}$	#n	#t	$\bar{c}$	#n	#t	$\bar{c}$	#n	#t	$\bar{c}$
bordeaux	32	19	189	16	10	83	20	58	141	0	0	0	20	114	235	62	76	306
lille	16	22	267	12	14	586	64	76	187	16	45	383	20	121	282	44	105	297
lyon	6	12	17	6	8	443	24	43	226	8	22	672	6	62	226	6	18	626
nancy	10	31	120	0	0	0	14	70	126	0	0	0	10	88	131	0	0	0
orsay	10	36	26	8	16	337	0	0	0	4	10	54	10	79	289	20	83	431
rennes	10	13	38	0	0	0	14	57	97	0	0	0	0	0	0	0	0	0
sophia	12	24	121	40	24	137	0	0	0	28	58	174	20	135	178	28	121	468
toulouse	20	27	219	12	12	639	20	60	249	20	29	586	20	125	374	0	0	0
TOTAL	116	184		94	84		156	364		76	164		106	724		160	403	

5.1. Experiment Setup

Workflow Configuration: We executed the IR workflow using 40, 20, 10, and 2 subjects. By varying the number of subjects used we calculated the makespan of the workflow, total storage space required for execution, and parallelism that can be achieved. We grouped the tasks when there was more than a single sequential task between two synchronizing tasks, as depicted in Figure 2. Grouping tasks implicitly demands more than one task to be executed at the same site where it is submitted, unlike the ungrouped version where all tasks would be distributed.

Resource Configuration: We used the resources provided by Grid'5000 as depicted in Figure 11. The Grid'5000 project provides a highly reconfigurable, controllable, and monitorable experimental Grid platform gathering 9 sites geographically distributed in France

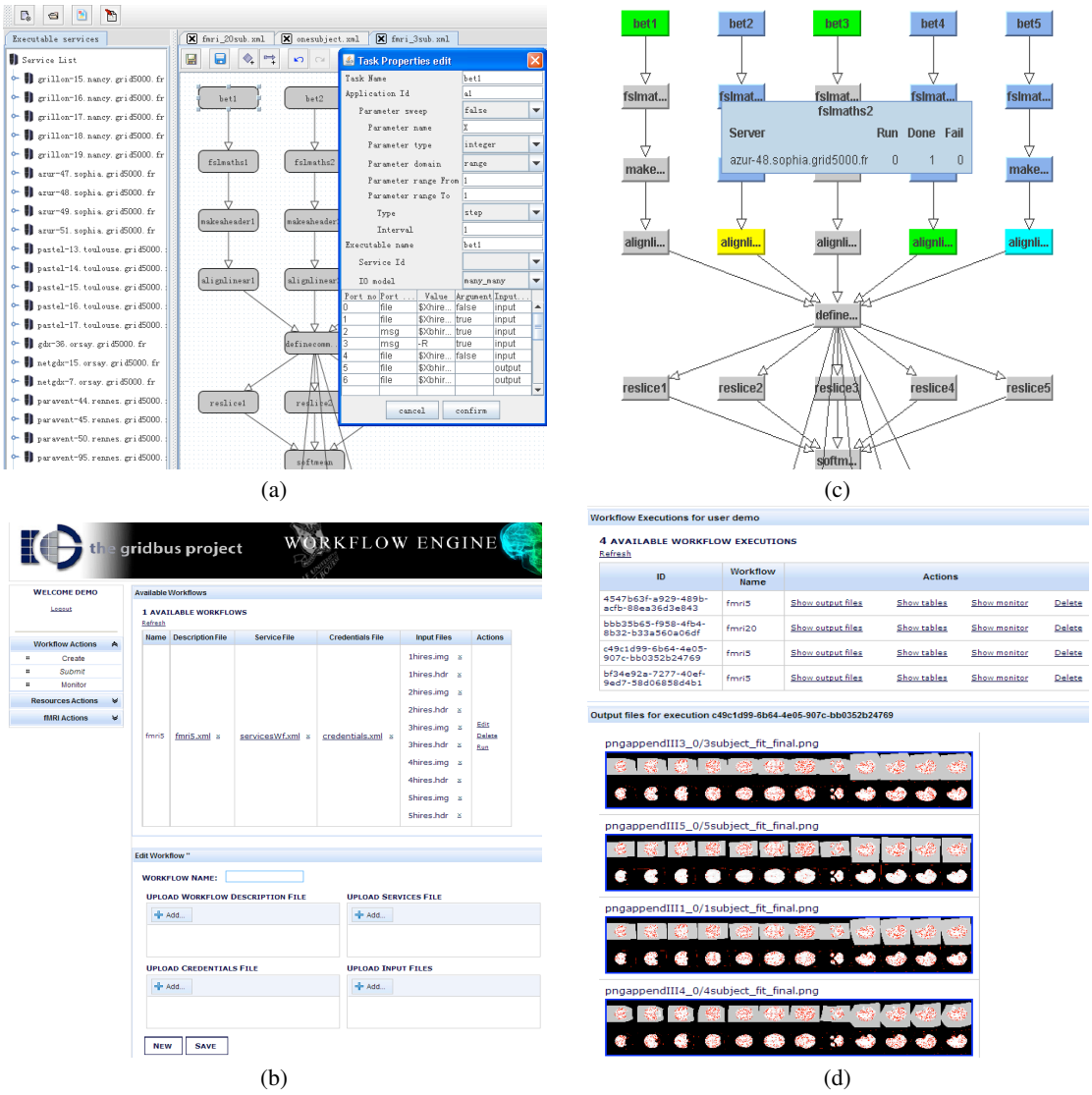


Figure 8. Grid Portal. (a) Workflow editor. (b) The submission interface. (c) Workflow monitor showing status of tasks in colors (cyan = ready, yellow = submitting, green = executing, blue = done). (d) Multiple workflow runs and output.

featuring a total of 5000 processors [5]. Table II lists the Grid'5000 sites used for the experiment. The resources were reserved for the duration of the experiment. The reservation ensured that the Grid resources were dedicated to our experiment. We used resources with the 'x86_64' CPU architecture with AMD Opteron Processors-246, 248, 250, 252, and 275. We used 8 out of the 9 sites (excluding Grenoble). The distributed resources across 8 sites have varying network

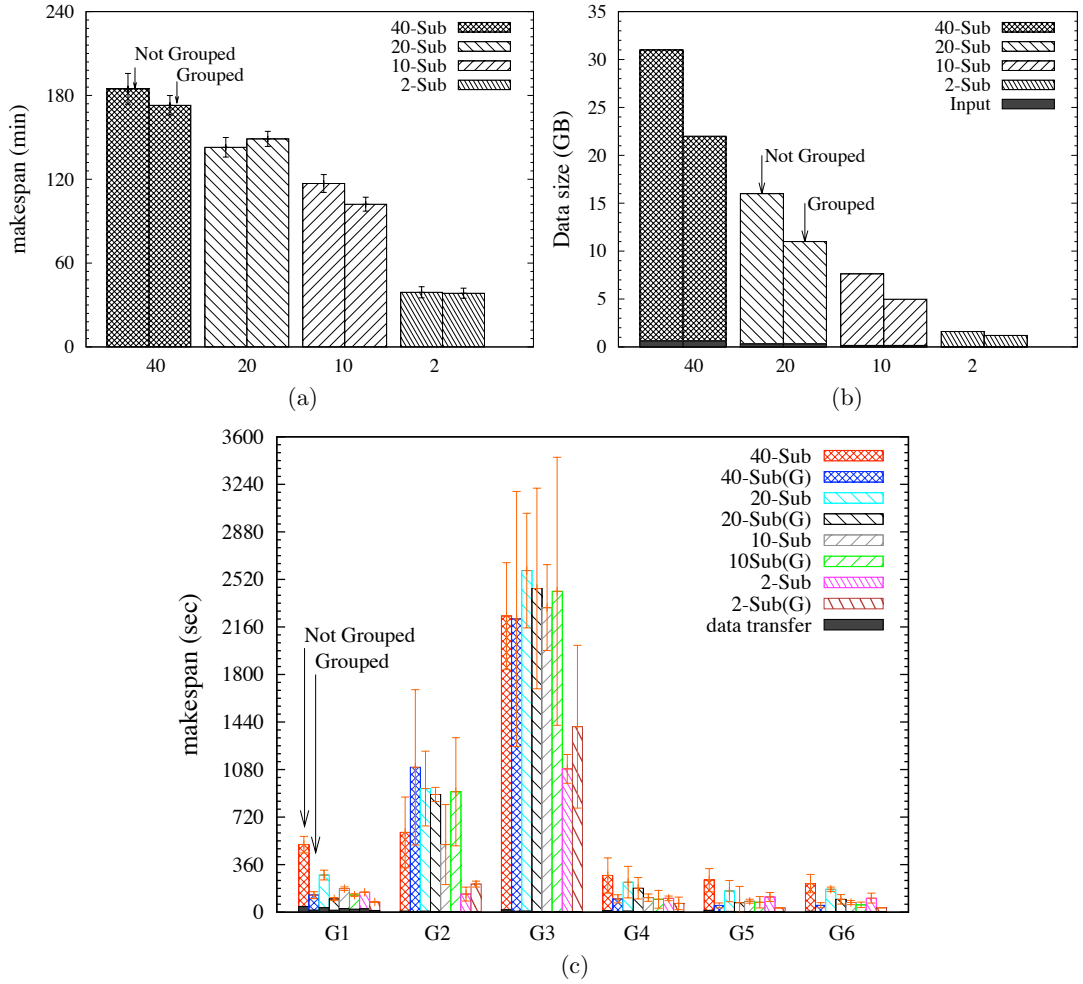


Figure 9. (a) Comparison of makespan of workflow according to the number of subjects used. (b) Data size according to the number of subjects used. (c) Comparison of makespan between grouped and ungrouped tasks (see Figure 2 for grouping of tasks).

interconnection bandwidth, number of cores per CPU, CPU frequency, memory, and storage space available [5].

The characteristics of Grid'5000 resources does not vary across 9 sites so much to abruptly affect our application performance. Also, Grid'5000 mandates the users to reserve the necessary nodes before execution. This scenario led us to use a basic level scheduling algorithm as listed in Algorithm 1.

Performance Metrics: As a measure of performance, we used average makespan as the primary metric. Makespan of each workflow is measured by taking the difference between the

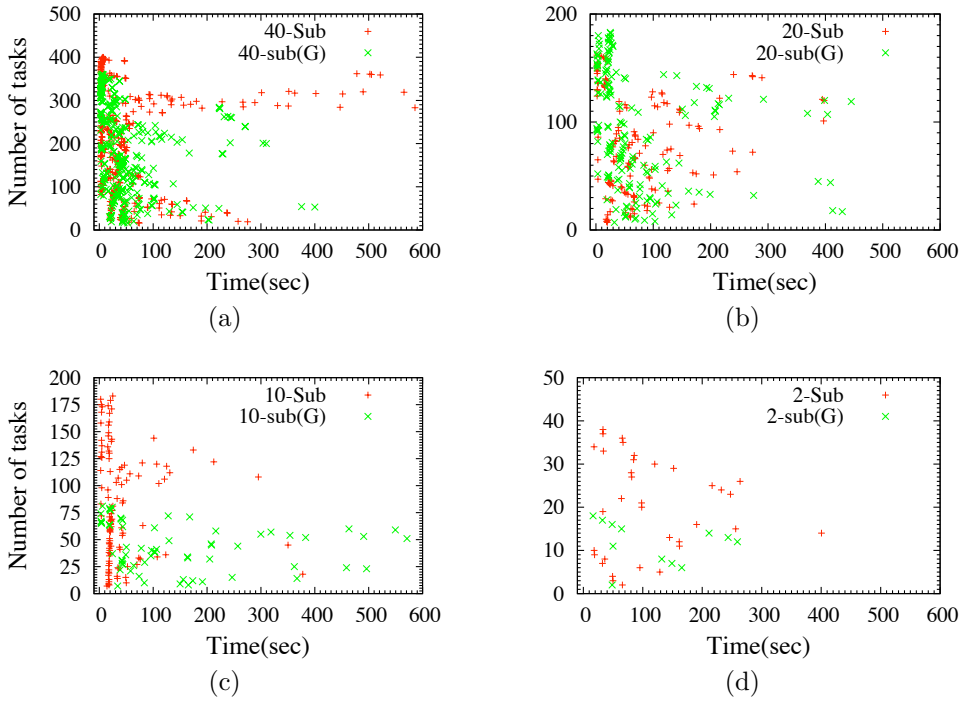


Figure 10. Number of tasks executed vs. Time: Parallelism that was achieved in the system. (a) Number of tasks executed in time for 40 subjects. (b) Number of tasks executed in time for 20 subjects. (c) Number of tasks executed in time for 10 subjects. (d) Number of tasks executed in time for 2 subjects.

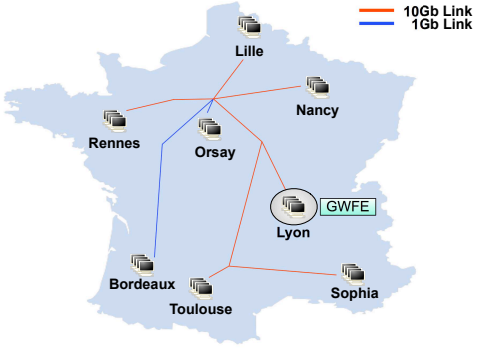


Figure 11. Grid'5000 Network [5].

submission time of the first submitted task and the output arrival time of the last exit task to be executed on the system. Makespan also includes the staging-in of the input files to the entry tasks and the staging-out of the results from the exit tasks.

5.2. Results and Observations

Table II lists the number of cores used at each site, the number of tasks submitted to the site and the average computation time used at the site for each experiment group. Figure 5 depicts the total makespan for different subjects, comparison of makespan between grouped and ungrouped tasks of the workflow and the size of data produced during the execution. FIgure 5 depicts parallelism of tasks executed in time by the GWFE for 40, 20, 10 and 2 subjects.

Execution of the IR workflow on the Grid showed significant advantages over a single machine. The total makespan of the workflow decreased considerably from 2.5 days in a single machine to approximately 3 hours on the Grid. The storage requirements were distributed among the resources used. As the number of subjects used was increased, the makespan increased slightly. This can be attributed to the increase in execution time of synchronizing tasks and the coordination time required by the system for additional tasks. The main point to be noted is that as the number of subjects was increased, the average makespan remained within similar bounds and did not increase exponentially, as can be seen for 40, 20, and 10 subjects in Figure 5 (a). By inference for more than 40 subjects the makespan should not increase by more than double the difference between the 40 subject and 20 subject makespan.

Grouping of tasks reduced the transmission of data between individual tasks as they were executed on the same machine the group was submitted to. Also, no coordination was required for the individual tasks in the group. This contributed to the reduced makespan in the case of grouped tasks. Figure 5(c) shows that the grouping of tasks that have higher value of standard deviation of execution did not yield an optimized makespan. Ungrouping tasks with higher execution time and a higher standard deviation value gave lower makespan than the grouped version (center of the graph) of that set of tasks. Tasks with lower execution time and lower standard deviation value had lower makespan value when grouped than when ungrouped.

The size of data produced during the execution of the workflow increased when the number of subjects was increased. The input data size (16MB per subject) was low in comparison to the total data produced during the execution as shown in Figure 5(b).

Due to the use of highly available resources, almost all the workflow's ready tasks were executed in parallel, as depicted in Figure 5. The graph shows the plot of tasks that finished execution versus time. At a certain interval in the beginning of execution most of the tasks finished execution at the same time showing the parallelism of execution of tasks. Most of the grouped tasks finished execution at the beginning of the execution interval unlike ungrouped tasks. This early execution behaviour helped reduce the makespan of the whole workflow as the grouped tasks executed more than one task at a time through a bash script, which is seen as a single task by the resource. In the case of ungrouped tasks each task needed to be mapped onto a resource and as the resource approached its maximum job limit, no more tasks could be submitted to it. This is also the reason that fewer grouped tasks were executed on the system than ungrouped tasks after 100 seconds.

We used a just-in-time scheduling algorithm to schedule tasks in the workflow, as listed in Algorithm 1. As the tasks became ready the scheduler was able to find the available resource and then submitted the task to it for execution. Failure of tasks was handled on a per task basis. When tasks failed, they were re-submitted to another resource, which did not have a failure history for those tasks. Although some tasks failed to execute and were rescheduled,

their total count was very small. Tasks can fail due to many reasons. In our experiment, failures occured when a task finishes execution without throwing any errors but the data produced by the task is not complete. In such cases, the child tasks that depend on that data always fail. We can resolve this fault by resubmitting the immediate parent task and all its child tasks for execution.

The workflow was executed multiple times by changing the parameters of the workflow with the help of the Grid Portal. This feature provided flexibility while executing grouped and ungrouped versions of the workflow for each of the 40,20,10 and 2 subjects. Without this feature, orchestrating the whole experiment would have taken a longer amount of time than executing the application on a single machine.

6. CONCLUSION AND FUTURE WORK

In this work, we presented the processing of a compute and data-intensive brain imaging application in the form of a workflow on the Grid. We implemented the components in the context of executing Image Registration (IR) for fMRI studies. We used the Gridbus Workflow Engine as workflow management system and the Gridbus Broker as the mediator to access the distributed resources. We have described in detail all the components of our system. Our experiments demonstrated that the IR procedure can have significantly reduced makespan, greater distribution of storage space and increased flexibility when executed on the Grid. Our analysis and the results of this neuroscience application show that there exists a greater motive and higher potential in strengthening collaboration between eScience communities and industry.

As part of our continuing efforts, we are enhancing the GWFE to support SLA based workflow scheduling. We are also working on scheduling algorithms that minimize usage of bandwidth for data-intensive workflow applications yet maintaining users' quality of service. The scheduling algorithms should be more generic and should consider Grid resources whose performance and availability are dynamically changing. Also, for workflows with large number of tasks, it is necessary to checkpoint the states of tasks and track provenance of data so that failures can be handled gracefully without repeating the execution of completed tasks. We could also provide mechanisms for changing the structure of the workflow at run-time. The exploration of these optimizations techniques and flexibility to the user is an important area of future research.

ACKNOWLEDGEMENTS

This work is partially supported through Australian Research Council (ARC) Discovery Project grant, and International Science Linkages (ISL) program of the Department of Innovation, Industry, Science and Research. Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research. We thank Dr. Jia Yu from the University of Melbourne for her efforts in building the Gridbus Workflow Engine. We would also like to thank Alexandre di Costanzo and Srikumar Venugopal from the University of Melbourne for their help in setting up Grid resources for our experiment.

REFERENCES

[1] AIR. Automated image registration, 2002. URL `http://bishopw.loni.ucla.edu/AIR5/index.html`.

[2] Kaizar Amin, Gregor von Laszewski, Mihael Hategan, Nestor J. Zaluzec, Shawn Hampton, and Albert Rossi. Gridant: A client-controllable grid work.ow system. In *HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 7*, 2004.

[3] Rajkumar Buyya, Susumu Date, Yuko Mizuno-Matsumoto, Srikumar Venugopal, and David Abramson. Neuroscience instrumentation and distributed analysis of brain activity data: a case for escience on global grids. *Concurrency and Computation : Practice & Experience*, 17, 2005.

[4] Junwei Cao, Stephen A. Jarvis, Subhash Saini, and Graham R. Nudd. Gridflow: Workflow management for grid computing. In *CCGRID '03: Proceedings of the 3st International Symposium on Cluster Computing and the Grid*, 2003.

[5] Franck Cappello and Henri Bal. Toward an international "computer science grid". In *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, pages 3–12, Washington, DC, USA, 2007. IEEE.

[6] Jinjun Chen and Yun Yang. A taxonomy of grid workflow verification and validation. *Concurrency and Computation: Practice and Experience*, 20(4):347–360, 2008. ISSN 1532-0626.

[7] Ewa Deelman, James Blythe, and et al. Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing*, 1, 2003.

[8] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G. Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, and Daniel S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Sci. Program.*, 13(3):219–237, 2005. ISSN 1058-9244.

[9] Randy E. Ellis and Terry M. Peters. Medical image computing and computer-assisted intervention. In *MICCAI (2)*, 2003.

[10] Christian Barillot et al. Neurobase: Management of distributed and heterogeneous information sources in neuroimaging. In *Didamic Workshop, MICCAI 2004 Conference*, 2004.

[11] Ian T. Foster, Jens-S. Vöckler, Michael Wilde, and Yong Zhao. Chimera: Avirtual data system for representing, querying, and automating data derivation. In *SSDBM '02: Proceedings of the 14th International Conference on Scientific and Statistical Database Management*, 2002.

[12] Nathalie Furmento, William Lee, Anthony Mayer, Steven Newhouse, and John Darlington. Iceni: an open grid service architecture implemented with jini. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, 2002.

[13] Yolanda Gil, Ewa Deelman, Mark Ellisman, Thomas Fahringer, Geoffrey Fox, Dennis Gannon, Carole Goble, Miron Livny, Luc Moreau, and Jim Myers. Examining the challenges of scientific workflows. *Computer*, 40(12), 2007.

[14] 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, November 2004.

[15] Silvia D. Olabarriaga, Piter T. de Boer, Ketan Maheshwari, Adam Belloum, Jeroen G. Snel, Aart J. Nederveen, and Maurice Bouwhuis. Virtual lab for fmri: Bridging the usability gap. In *E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, 2006.

[16] Russell A. Poldrack and Joseph T. Devlina. On the fundamental role of anatomy in functional imaging: Reply to commentaries on in praise of tedious anatomy. *NeuroImage*, 37, 2007.

[17] David E. Rex, Jeffrey Q. Ma, and Arthur W. Arthur W. Toga. The loni pipeline processing environment. *NeuroImage*, 19, 2003.

[18] Mathilde Romberg. The unicore architecture: Seamless access to distributed resources. In *HPDC '99: Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing*, 1999.

[19] S.M. Smith, M. Jenkinson, M.W. Woolrich, C.F. Beckmann, T.E.J. Behrens, H. Johansen-Berg, P.R. Bannister, M. De Luca, I. Drobnjak, D.E. Flitney, R. Niazy, J. Saunders, J. Vickers, Y. Zhang, N. De Stefano, J.M. Brady, and P.M. Matthews. Advances in functional and structural mr image analysis and implementation as fsl. NeuroImage, 23(S1):208-219, 2007.

[20] Ian Taylor, Matthew Shields, Ian Wang, and Roger Philp. Distributed p2p computing within triana: A galaxy visualization test case. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, 2003.

[21] Srikumar Venugopal, Rajkumar Buyya, and Lyle Winton. A grid service broker for scheduling e-science applications on global data grids: Research articles. *Concurrency and Computation: Practice & Experience*, 18(6):685–699, 2006. ISSN 1532-0626. doi: http://dx.doi.org/10.1002/cpe.v18:6.

[22] Jia Yu and Rajkumar Buyya. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Record*, 34(3):44–49, 2005. ISSN 0163-5808.

[23] Jia Yu and Rajkumar Buyya. A novel architecture for realizing grid workflow using tuple spaces. *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*, 2004.

[24] Yong Zhao, Jed Dobson, Ian Foster, Luc Moreau, and Michael Wilde. A notation and system for expressing and executing cleanly typed workflows on messy scientific data. *SIGMOD Record*, 34 (3):37–43, 2005. ISSN 0163-5808. doi: http://doi.acm.org/10.1145/1084805.1084813.

