Brain Image Registration Analysis Workflow for fMRI Studies on Global Grids

Suraj Pandey, William Voorsluys, Mustafizur Rahman, Rajkumar Buyya
Grid Computing and Distributed Systems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{spandey, williamv, mmrahman, raj}@csse.unimelb.edu.au
James Dobson
Department of Psychological & Brain Sciences, Dartmouth College, USA
james.e.dobson@dartmouth.edu
Kenneth Chiu
Grid Computing Research Lab, Department of Computer Science
State University of New York (SUNY) at Binghamton, NY, USA
kchiu@cs.binghamton.edu

Abstract

Scientific applications like neuroscience data analysis are usually compute and data-intensive. With the use of globally 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 transform it to a workflow. 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 global Grids.

1 Introduction

Nowadays, many scientific experiments such as structural biology and chemistry, neuroscience data analysis, disaster recovery, etc., are conducted through complex and distributed scientific computations that are represented and structured as scientific workflows [11]. Scientific workflows usually need to process huge amount of data and 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. 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 the 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.

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) [18], is one such Grid-based workflow management system that aids 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 IR process for fMRI applications. We first characterize the application and identify its requirements. We then execute it on global Grids with the help of GWFE 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.

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

2 Related Work

Several projects are investigating workflow and Grid scheduling techniques. We categorize these under **application** and **Workflow Management Systems**.

Application: Olbarriaga et al. [13] 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 [8] 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 described the composition and on-demand deployment of the neuroscience application as a parameter-sweep model on global Grids. They also designed and developed a Magnetoencephalography (MEG) data analysis system.

Ellis et al. [7] 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.

The LONI Pipeline [15] was developed to facilitate ease of workflow construction, validation and execution like many similar workflow environments, primarily used in the context of neuroimaging.

The NIMH Neuroimaging Informatics Technology Initiative (NIFTI[1]) was formed to aid in the development and enhancement of informatics tools for neuroimaging. Likewise, the Biomedical Informatics Research Network (BIRN[2]) is another high profile effort working to develop standards (eg. LONI) among its consortia members.

[1]http://nifti.nimh.nih.gov
[2]http://nbhirn.net

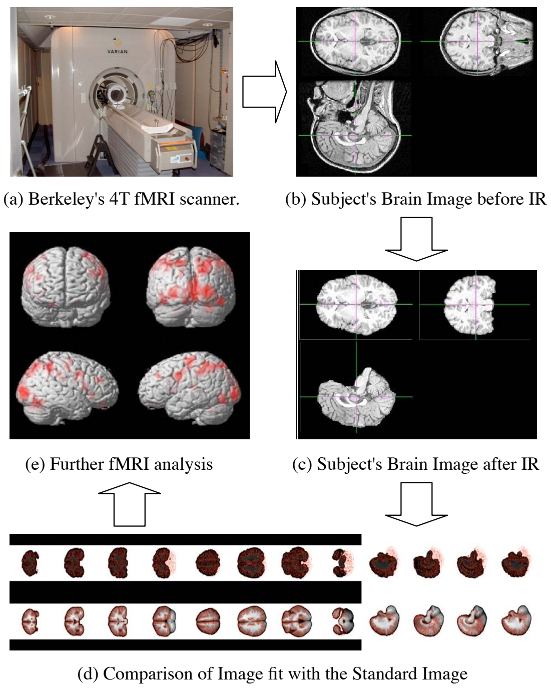

(a) Berkeley's 4T fMRI scanner. (b) Subject's Brain Image before IR (c) Subject's Brain Image after IR (d) Comparison of Image fit with the Standard Image (e) Further fMRI analysis

Figure 1: Image Registration and fMRI.

Workflow Management Systems: Deelman et al. [6] have done considerable work on planning, mapping and data-reuse in the area of workflow scheduling. They propose Pegasus [6], 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 [9] and Pegasus based mapping, it can execute complex workflows based on pre-planning. The Taverna project [12] has developed a tool for the composition and enactment of bioinformatics workflows for the life science community. It provides GUI for the composition of workflows. Other well-known projects on workflow systems include GridFlow [4], ICENI [10], GridAnt [2] and Triana [16].

3 Scientific Workflow Applications: A Scenario and Requirements

We describe the IR process for fMRI studies as a workflow application. We construct the IR workflow and describe each of its tasks, and tabulate the execution requirements of 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 the Magnetic Resonance (MR) scanner, would be asked to perform a task, 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 the process of estimating an optimal transformation between two images, also known as "Spatial Normalization" in functional neuroimaging [14]. 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 movement 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: Figure 2 shows the IR procedure expressed as a workflow. Tasks in the workflow are linked according to their data dependencies. Individual tasks in the workflow are described below [1].

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

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. It generates *.air* files that can be used to reslice data set to match a standard data set. We use affine 12-parameter model.

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

RESLICE: takes a *.air* file, uses it to load an image file and generates a new, realigned file.

SOFTMEAN: averages together a series of files.

ALIGN_WARP: is a nonlinear registration tool that generates a *.warp* file that can be used to reslice data set to match a standard data set.

RESLICE_WARP: takes a *.warp* file, uses it to load the corresponding image file and generates a new, realigned file.

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

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

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

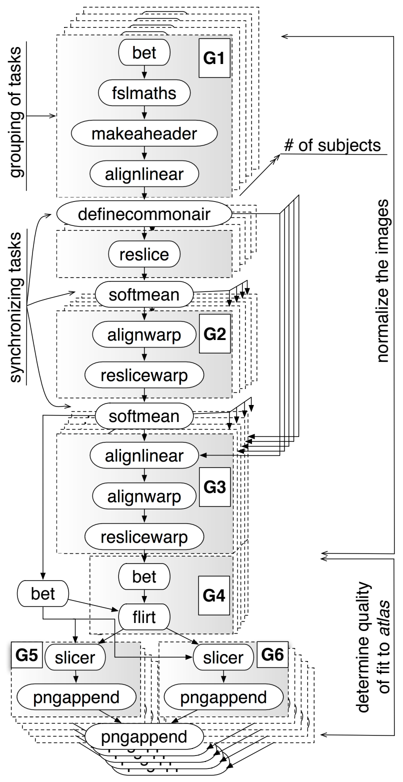


Figure 2: Image Registration Workflow.

Application Requirements: According to Zhao et.al [19], 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.

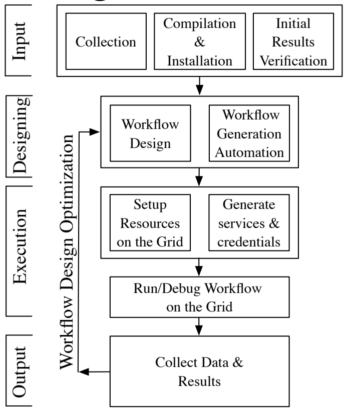
Table 1 lists each task, its input files and sizes, its average computation time ($\bar{w_i}$) on a single machine, and standard deviation (σ) computed over 40 subjects on a set of 10 random machines in Grid'5000 [5].

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

4 Workflow Management on the Grid



Input
Collection
Compilation & Installation
Initial Results Verification
Designing
Workflow Design
Workflow Generation Automation
Execution
Setup Resources on the Grid
Generate services & credentials
Run/Debug Workflow on the Grid
Output
Collect Data & Results
Workflow Design Optimization

Figure 3: Deployment Cycle.

The life cycle of deployment of the workflow is depicted in Figure 3. In the collection phase, scientists provide the input data, batch scripts, sample output files, and application requirements. In order to run the application on the Grid, 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 dataflow need to be designed. The generation of the workflow in terms of the workflow language used could to be automated. However, it should also take users' run-time parameters into account.

In the *Execution* phase, compute and data resources need to be setup, where the application can be executed. The resource list, its credentials and the services provided by each resource could be obtained from a catalogue. With changing resource availability and conditions, resource list is required to be updated dynamically. 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 model being used. Depending on the performance analysis of the execution, the design of a workflow can be further optimized.

We now describe the components of the Gridbus workflow management system.

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 4 (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 4 (d), 1 (d)).
4. A resource information page, which shows the characteristics of all available grid resources.
5. An application specific page, which in the current implementation, helps generate *.xml* file for the IR workflow 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.

Workflow Editor: The Gridbus workflow editor provides a Graphical User Interface (GUI) that enables users to create and modify existing workflows. The definition of the workflow is based on an XML-based workflow language (xWFL). Using the editor users design and create the workflows for complex scientific procedures by dragging and dropping boxes and arrows, connecting them, and defining their properties (input, output, parameters etc.). Boxes and directed arrows represent workflow tasks and data dependencies between them, respectively.

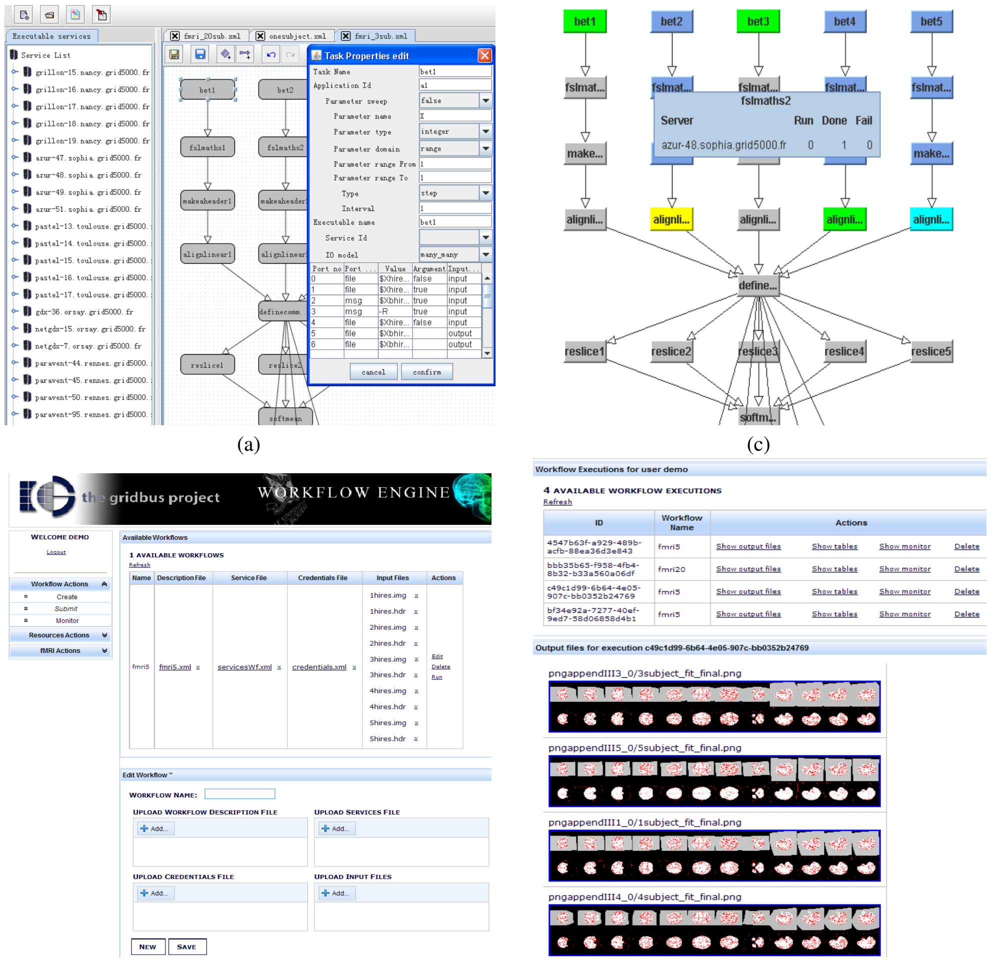
The GUI isolates users from the complexity of composing and editing XML entries in the files. However, for advance users, editor provides direct access to the XML details of the workflow design. As the workflow editor can be accessed through the workflow portal, users can create or edit and save the workflows both locally and on the server. Common editing operations such as cut, copy, paste can be performed on multiple workflows.

Workflow Monitor: The Gridbus Workflow Monitor provides a GUI for viewing the status of each task in the

Table 1: Characteristics of tasks for a single subject.

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 required for a single subject registration:		**558.2 MB**	**$\sim$ 69 min**	


Executable services

Service List
- grillon-15.nancy.grid5000.fr
- grillon-16.nancy.grid5000.fr
- grillon-17.nancy.grid5000.fr
- grillon-18.nancy.grid5000.fr
- grillon-19.nancy.grid5000.fr
- azur-47.sophia.grid5000.fr
- azur-48.sophia.grid5000.fr
- azur-49.sophia.grid5000.fr
- azur-51.sophia.grid5000.fr
- pastel-13.toulouse.grid5000.
- pastel-14.toulouse.grid5000.
- pastel-15.toulouse.grid5000.
- pastel-16.toulouse.grid5000.
- pastel-17.toulouse.grid5000.
- gdx-36.orsay.grid5000.fr
- netgdx-15.orsay.grid5000.fr
- netgdx-7.orsay.grid5000.fr
- paravent-44.rennes.grid5000.
- paravent-45.rennes.grid5000.
- paravent-50.rennes.grid5000.
- paravent-95.rennes.grid5000.

fmri_20sub.xml | onesubject.xml | fmri_3sub.xml

Task Properties edit

Task Name	bet1
Application Id	a1
Parameter sweep	false
Parameter name	X
Parameter type	integer
Parameter domain	range
Parameter range From	1
Parameter range To	1
Type	step
Interval	1
Executable name	bet1
Service Id	
IO model	many_many

Port no	Port ...	Value	Argument	Input...
0	file	$Xhire...	false	input
1	file	$Xhire...	true	input
2	msg	$Xbhir...	true	input
3	msg	-R	true	input
4	file	$Xhire...	false	input
5	file	$Xbhir...		output
6	file	$Xbhir...		output

(a)

the gridbus project WORKFLOW ENGINE

WELCOME DEMO

Logout

Workflow Actions: Create, Submit, Monitor

Resources Actions

fMRI Actions

Available Workflows

1 AVAILABLE WORKFLOWS

Refresh

Name	Description File	Service File	Credentials File	Input Files	Actions
fmri5	fmri5.xml	servicesWf.xml	credentials.xml	1hires.img, 1hires.hdr, 2hires.img, 2hires.hdr, 3hires.img, 3hires.hdr, 4hires.img, 4hires.hdr, 5hires.img, 5hires.hdr	Edit Delete Run

Edit Workflow ''

WORKFLOW NAME:

UPLOAD WORKFLOW DESCRIPTION FILE — Add...

UPLOAD SERVICES FILE — Add...

UPLOAD CREDENTIALS FILE — Add...

UPLOAD INPUT FILES — Add...

NEW SAVE

(b)

(c)

Workflow Executions for user demo

4 AVAILABLE WORKFLOW EXECUTIONS

Refresh

ID	Workflow Name	Actions			
4547b63f-a929-489b-acfb-88ea36d3e843	fmri5	Show output files	Show tables	Show monitor	Delete
bbb35b65-f958-4fb4-8b32-b33a560a06df	fmri20	Show output files	Show tables	Show monitor	Delete
c49c1d99-6b64-4e05-907c-bb0352b24769	fmri5	Show output files	Show tables	Show monitor	Delete
bf34e92a-7277-40ef-9ed7-58d06858d4b1	fmri5	Show output files	Show tables	Show monitor	Delete

Output files for execution c49c1d99-6b64-4e05-907c-bb0352b24769

pngappendIII3_0/3subject_fit_final.png

pngappendIII5_0/5subject_fit_final.png

pngappendIII1_0/1subject_fit_final.png

pngappendIII4_0/4subject_fit_final.png

(d)

Figure 4: 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.

workflow. Users can easily view the ready, executing, stage-in, and completed tasks as depicted in Figure 4. 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 failures for each task, if any. The workflow structure is editable such that users can drag tasks and separate tasks of interest when there are numerous tasks in the workflow.

Gridbus Workflow Engine: Scientific application portals submit the 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 the 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 [17]. GWFE uses Gridbus Broker for deploying and managing task execution on various middlewares as a dispatcher component. Gridbus Broker 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. In addition to random and round-robin scheduling policies, GWFE also has a Just-In-Time scheduling policy that allows the resource allocation decision to be made at the time of task submission in order to better adapt to the changing dynamic Grid environment.

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 '08, May 19-22, 2008, using resources provided by SUNY at Binghamton and the University of Melbourne. For this paper we executed the application on Grid'5000 [5]. We now describe the experiment setup, results, and observations.

5.1 Experiment Setup

We executed the IR workflow with 40, 20, 10, and 2 subjects. By varying the number of subjects, 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 one 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.

We used the resources provided by Grid'5000. The Grid'5000 project provides a highly reconfigurable, controllable, and monitorable experimental Grid platform gathering 9 sites geographically distributed in France featuring a total of 5000 processors [5]. Table 2 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 interconnection bandwidth, number of cores per CPU, CPU frequency, memory, and storage space available [5] .

As a measure of performance, we used average makespan as a metric. Makespan of each workflow is measured by taking the difference between the submission time of the first submitted task and the output arrival time of the last exit task to be executed on the system. This time 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 2 lists the number of cores used at each site, number of tasks submitted to a site and the average computation time at each site for each experiment. Figure 5 depicts the total makespan for different subjects, comparison of makespan between grouped and un-grouped tasks of the workflow, size of data produced during the execution, and parallelism of tasks executed in time by the GWFE.

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

Table 2: Grid5000 sites used; # cores, # tasks executed and average computation time ($\bar{c}$) taken on each site for each experimental group.

	Name	10Sub			10Sub (G)			20Sub			20Sub (G)			40Sub			40Sub (G)		
		#Cores	#Tasks	$\bar{C}$ (sec)	#Cores	#Tasks	$\bar{C}$ (sec)	#Cores	#Tasks	$\bar{C}$ (sec)	#Cores	#Tasks	$\bar{C}$ (sec)	#Cores	#Tasks	$\bar{C}$ (sec)	#Cores	#Tasks	$\bar{C}$ (sec)
Grid5000 Sites	bordeaux	32	19	189.09	16	10	83.27	20	58	140.63	0	0	0.00	20	114	235.48	62	76	305.95
	lille	16	22	267.36	12	14	586.20	64	76	187.09	16	45	382.61	20	121	282.03	44	105	297.35
	lyon	6	12	17.37	6	8	443.08	24	43	225.83	8	22	671.63	6	62	226.42	6	18	626.19
	nancy	10	31	120.50	0	0	0.00	14	70	125.62	0	0	0.00	10	88	131.01	0	0	0.00
	orsay	10	36	25.82	8	16	337.12	0	0	0.00	4	10	53.91	10	79	289.06	20	83	431.44
	rennes	10	13	38.20	0	0	0.00	14	57	97.41	0	0	0.00	0	0	0.00	0	0	0.00
	sophia	12	24	121.45	40	24	137.55	0	0	0.00	28	58	174.56	20	135	178.10	28	121	467.98
	toulouse	20	27	219.53	12	12	639.25	20	60	249.36	20	29	586.94	20	125	373.87	0	0	0.00
	TOTAL	116	184		94	84		156	364		76	164		106	724		160	403	

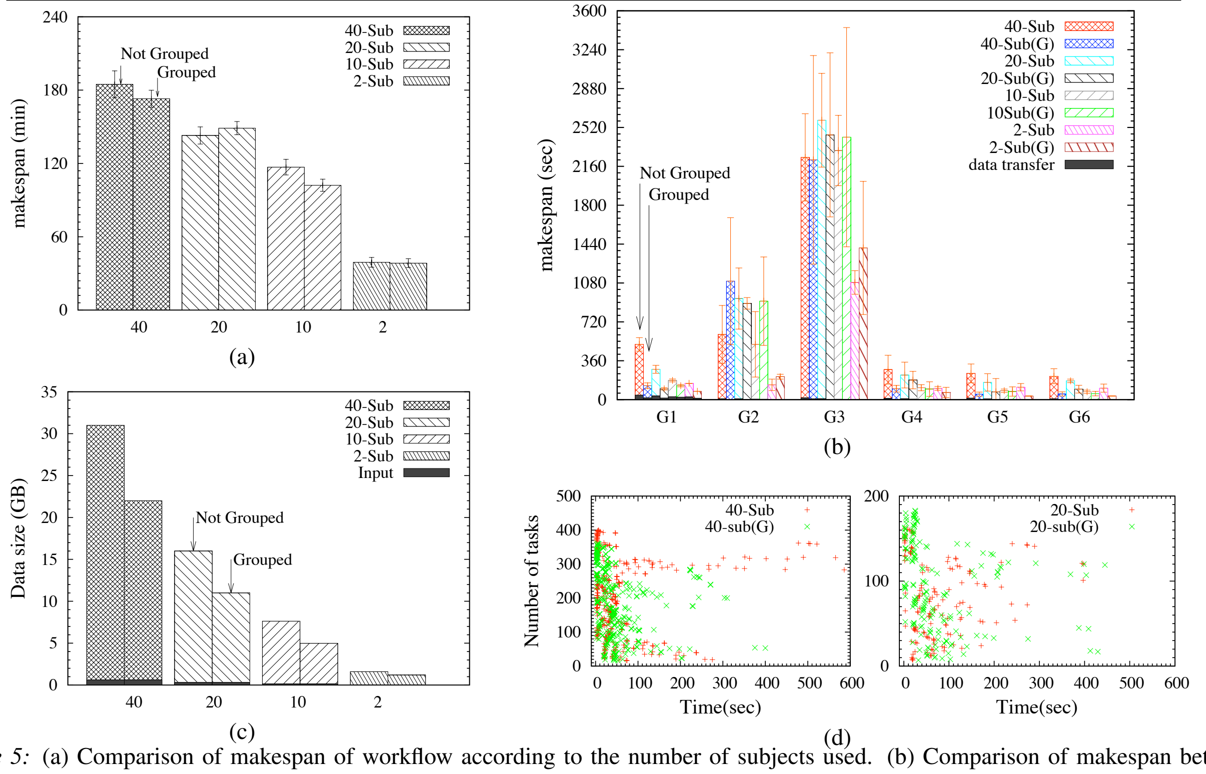


Figure 5: (a) Comparison of makespan of workflow according to the number of subjects used. (b) Comparison of makespan between grouped and ungrouped tasks (see Figure 2 for grouping of tasks). (c) Data size according to the number of subjects used. (d) # of tasks executed vs. Time: Parallelism that was achieved in the system.

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(b) 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(c).

Due to the use of highly available resources, almost all the workflow's ready tasks were executed in parallel, as depicted in Figure 5(d). 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 the tasks. As the tasks became ready the scheduler was able to find the best 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.

The workflow was executed multiple times by changing input parameters. 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 manually would have taken longer time than executing the application on a single machine.

6 Conclusions

In this work, we presented the processing of a compute and data-intensive scientific 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. 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 the efficient scheduling of compute and data-intensive scientific workflows on global Grids.

Acknowledgments

This work is partially supported through Australian Research Council (ARC) Discovery Project grant. Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research. We would also like to thank Alexandre di Costanzo and Srikumar Venugopal of the University of Melbourne.

References

[1] AIR. Automated image registration, 2002.

[2] K. Amin, G. von Laszewski, M. Hategan, N. J. Zaluzec, S. Hampton, and A. 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] R. Buyya, S. Date, Y. Mizuno-Matsumoto, S. Venugopal, and D. 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] J. Cao, S. A. Jarvis, S. Saini, and G. R. Nudd. Gridflow: Workflow management for grid computing. In *CCGRID '03: Proceedings of the 3st International Symposium on Cluster Computing and the Grid*, pages 198–205, Washington, DC, USA, 2003.

[5] F. Cappello and H. 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] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Sci. Program.*, 13(3):219–237, 2005.

[7] R. E. Ellis and T. M. Peters. Medical image computing and computer-assisted intervention. In *MICCAI (2)*, 2003.

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

[9] I. T. Foster, J.-S. Vöckler, M. Wilde, and Y. 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.

[10] N. Furmento, W. Lee, A. Mayer, S. Newhouse, and J. Darlington. Iceni: an open grid service architecture implemented with jini. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, 2002.

[11] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers. Examining the challenges of scientific workflows. *Computer*, 40(12), 2007.

[12] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, November 2004.

[13] S. D. Olabarriaga, P. T. de Boer, K. Maheshwari, A. Belloum, J. G. Snel, A. J. Nederveen, and M. 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.

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

[15] D. E. Rex, J. Q. Ma, and A. W. Arthur W. Toga. The loni pipeline processing environment. *NeuroImage*, 19, 2003.

[16] I. Taylor, M. Shields, I. Wang, and R. 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.

[17] S. Venugopal, R. Buyya, and L. 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.

[18] J. Yu and R. Buyya. A novel architecture for realizing grid workflow using tuple spaces. *Proc. of the Fifth IEEE/ACM International Workshop on Grid Computing*, 2004.

[19] Y. Zhao, J. Dobson, I. Foster, L. Moreau, and M. Wilde. A notation and system for expressing and executing cleanly typed workflows on messy scientific data. *SIGMOD Record*, 34(3):37–43, 2005.