Technical Report WM-CS-2009-05

College of
William & Mary
Department of Computer Science

WM-CS-2009-05

Non-Rigid Registration for Image-Guided Neurosurgery on the TeraGrid: A Case Study

Andriy Fedorov, Benjamin Clifford, Simon K. Warfield,
Ron Kikinis, Nikos Chrisochoides

April 24, 2009

Non-Rigid Registration for Image-Guided Neurosurgery on TeraGrid: A Case Study*

Andriy Fedorov[1] Benjamin Clifford[2] Simon K. Warfield[3]
Ron Kikinis[4] Nikos Chrisochoides[1]
[1]Center for Real-Time Computing, College of William and Mary
[2]Computation Institute, University of Chicago
[3]Children's Hospital, Boston [4]Brigham and Women's Hospital, Boston

Abstract

We present a feasibility study of using TeraGrid resources for computing non-rigid registration (NRR) of brain MRI. The end-to-end NRR procedure we consider has previously undergone clinical validation at Brigham and Women's Hospital, Boston. We consider the use of TeraGrid resources in two scenarios. First, we evaluate the feasibility of using a grid-enabled implementation to provide timely execution of the most time-consuming components of the registration. Second, we describe a workflow implementation to enable speculative computation of registration to improve confidence in the result and assist in retrospective evaluation of the method. Our initial results indicate that TeraGrid provides practical and cost-effective means to support IGNS image processing. At the same time, performance limitations of the intra-operative speculative NRR execution using conventional means of workflow scheduling prevent its intra-operative applications. The results of this paper motivate future work in high-throughput scheduling and execution models on TeraGrid, which can be leveraged by the workflow implementation we present.

1 Introduction

This paper has two main goals. Our first goal is to use distributed grid computing resources to provide computational platform for image processing during image-guided neurosurgery (IGNS). The second goal is to facilitate large experimental studies of image processing algorithms to improve our understanding of their behavior under different inputs. Non-rigid registration (NRR) can provide critical information to assist the surgeons during IGNS. However, before the specific NRR method can be used during the surgery, it must be throughly evaluated for a large number of retrospective data sets. Both of these tasks (time-constrained intra-operative NRR and large population retrospective studies) require non-trivial computing resources. We consider TeraGrid [6] as a provider of such resources.

Image-guided neurosurgery is concerned with the maximal removal of tumor while incurring minimum possible damage to the surrounding healthy tissue. Medical image data, like MRI, acquired pre-operatively, allows the identification the tumor and critical brain regions with high precision prior to the neurosurgery. However, as the tumor resection is

*This work was supported in part by NSF grants CSI-0719929 and CNS-0312980, and by John Simon Guggenheim Memorial Foundation.

progressing, brain deformation is unavoidable. Shift of the brain tissue invalidates preoperative data, and requires additional processing to account for this deformation. One of the available approaches to this problem is to acquire sparse images during IGNS, and update the preoperative data according to the deformation observed in the intra-operative images. The specific image processing operation that aligns two images using high order transformation is known as non-rigid registration.

Our efforts have been focused on the specific NRR method proposed by Clatz et al. [9]. In our earlier work [7], we described an implementation of this method, that makes it feasible to complete non-rigid registration using volumetric intra-operative data within the time constraints of the neurosurgery. The implementation we developed has been used routinely during neurosurgeries, and facilitated prospective evaluation of the method by Archip et al. [4]. In this paper we extend our previous work [7] in several directions:

- We present a loosely-coupled workflow implementation of the NRR procedure using SwiftScript [30]. There are many advantages of Swift over the customized shell scripts we used in [7]: easy integration of the new computational resources for workflow execution, automatic fault-tolerance and load-balancing on the task level, and flexible workflow submission and scheduling mechanisms facilitated by CoG Karajan [27].

- We suggest modifications to the current IGNS NRR protocol to leverage the distributed computing resources and estimate some of the sub-optimal NRR parameter values. This computation can be completed before the time-critical part of the procedure, and can potentially improve the accuracy of the NRR.

- We integrate intra-operative assessment [4, 13] within the NRR workflow, and consider speculative execution [19] of NRR to improve confidence in the results and achieve better accuracy of the solution.

The contribution of this paper is a portable implementation of the NRR workflow and a feasibility study of its execution on the TeraGrid resources. We evaluate intra-operative and retrospective scenarios of using this implementation, and suggest directions for future work to improve its performance on TeraGrid. The lessons learned from this evaluation help to identify specific areas in workflow scheduling that require improvements to facilitate high throughput for the broader class of time-critical applications. The TeraGrid resources provide the potential to enable solution of otherwise infeasible problems. As we show in this paper, the existing technology that enables access to these resources (e.g., Globus Toolkit, CoG Karajan and Swift) facilitates migration of legacy applications like NRR on the grid, although more work is required to fully benefit from TeraGrid capabilities.

2 Grid computing and medical applications

Since the introduction of the idea of a computational grid by Foster and Kesselman [16], significant effort has been focused on realization of their vision. Community efforts have been directed towards development of the supporting standards and software, deployment of production grid systems worldwide and porting applications on those systems [24]. One

of the most prominent such production system under continuous improvement and development is the USA-based TeraGrid [6]. As of May 2007, TeraGrid was *"...the world's largest, most comprehensive distributed cyber-infrastructure for open scientific research"*. Currently, TeraGrid connects 11 computational centers providing cumulative peak performance of 1124 teraflops. The capabilities of TeraGrid are continuously growing, providing computational and storage resources otherwise unavailable to any single research institution worldwide.

Concurrently with the development of the grid technology, we witness significant advances in medical image acquisition, analysis and visualization in the last decades. Medical images of high resolution and of different modalities are essential in prevention, diagnosis and treatment of many diseases. As the related technology is improving and clinical studies complete, the demand for analysis of the medical data will be growing. Distributed computational and storage resources, like those available via the TeraGrid, provide a promising platform to meet such demands. Medical applications that require processing and archival of large amounts of data contributed by geographically distributed medical centers were naturally some of the first to embrace grid computing. Representative projects are MammoGrid (maintenance and analysis of mammography data; Europe) [3] and Biomedical Informatics Research Network (BIRN) (analysis of functional MRI; USA) [1]. The infrastructures developed by these projects (by design) are not general to support image processing application like ours.

In the context of our project, we are interested in using a *general purpose* research grid environment for a specific medical application. This idea has been evaluated in the past for different medical applications. Dong et al. [11] present results of modeling the whole human arterial tree on TeraGrid. The Grid Enabled Neurosurgical Imaging Using Simulation (GENIUS) project [22] targets modeling and visualization of cerebral blood flow within 15-30 minutes to assist in surgical planning. The common feature of these two applications is that they show linear scalability up to thousands of cores. The absence of a global TeraGrid resource management system makes it almost impossible to use a large number of nodes at different sites for cross-site runs without prior arrangements. The aforementioned projects circumvented this problem by using the advance reservation capabilities available at some TeraGrid sites. However, advance reservations are not generally favored by the site administrators, as they tend to reduce the machine utilization. This is possibly the primary reason why advance reservations are not available at most of the TeraGrid sites. Moreover, in the case of intra-operative processing for IGNS, the actual time of the surgery is not known precisely well in advance. The surgery may also be canceled or postponed due to last minute considerations, e.g., changes in the patient's condition. The GENIUS project [22] considered the use of SPRUCE [5], an urgent computing system under development on TeraGrid. This approach is more practical, as it allows to dynamically affect the scheduling decisions to favor selected jobs. To the best of our knowledge, only one site of the TeraGrid (UC/ANL) provides SPRUCE-directed job preemption, and at some other limited number of sites SPRUCE can be used to schedule the SPRUCE jobs to run first, once the resources become available.

Non-rigid image registration is usually amenable to parallelization. However, its scalability is highly dependent on the specific algorithm used, and the NRR methods known to us [14, 7, 29] do not scale beyond hundreds of nodes. Still, such computational resources are not available locally at most hospitals, and a number of groups studied how the grid resources specifically can be used to increase the performance of medical image processing

computations. Lippman and Kruggel [20] present a framework for intra-operative image registration on the grid. They use a customized grid environment (GEMSS [2]), and do not evaluate the procedure on a general research grid, like TeraGrid. Stefanescu et al. [26] describe a parallel implementation of non-rigid registration deployed on a cluster of workstations as a web service (we evaluated this approach in [7]). Gropp et al. [18] study rigid registration, which is a computationally simpler problem, and discuss its different aspects in the context of grid computing. Glatard et al. [17] present a grid-enabled service-based infrastructure for comparing the performance of rigid registration implementations. Their work does explore the use of general-purpose grid resources, and uses workflow model for describing and scheduling the execution. However, their evaluation is done for the European grid infrastructure. The approaches to grid application development and workflow definition are different from those that can be applied on TeraGrid, and cannot be directly used in our application. In general, we observe that the studies of the NRR methods that are used in practice do not leverage general purpose grid systems.

In this paper we evaluate the capabilities of a general purpose research grid, TeraGrid, in enabling intra-operative, time-critical, non-rigid registration. The closest related work has been reported by Majumdar et al. [21], who studied the potential effects of network latency and batch scheduler on the total time required to compute non-rigid registration during image-guided neurosurgery. However, their studies are limited to experimentation with the solver component of the NRR procedure: they did not propose a grid-enabled end-to-end implementation of the NRR, and did not evaluate it within the current service-oriented framework [15] of the TeraGrid. On the contrary, we deploy and evaluate the complete end-to-end NRR method, which has been studied previously during IGNS [7, 4].

3 Non-rigid registration of brain MRI for IGNS

The details of the robust non-rigid registration method are given by Clatz et al. in [9]. Here we summarize the main points of the approach. Registration consists of the initialization, which can be done preoperatively, and the intra-operative part of computation. The goal of the registration is to enable enhanced visualization of the brain under intra-operative deformation. Thus, the preoperative (floating) image is deformed to match the intra-operative (reference) scan. An intra-operative image is acquired (thus, registration of the preoperative data is necessary) in the following cases: immediately after the patient head is fixed for the surgery (first intra-operative scan), after the skull opening, any time when the surgeon suspects significant shift of the brain, or there is a need to verify residual tumor volume.

Pre-operative Processing Multi-modal high resolution scans are acquired prior to the surgery to identify the tumor and critical regions in its vicinity, and to develop the appropriate resection strategy. The intra-cranial cavity (ICC) is segmented, and the patient-specific biomechanical model is constructed for the subsequent application of Finite Element Method (FEM). The registration algorithm is initialized with volumetric block matching between the floating and reference images at certain points of the image (registration points), which we describe next. The registration points are selected to be the locations of the image with the highest intensity variance in the surrounding region.

Intra-operative Processing Rigid alignment of preoperative data to the first intra-operative scan is the first processing task. This part of the intra-operative computation is not time-critical: usually, there is sufficient delay between the first intra-operative scan, and the skull opening. The time-critical component of the computation is initiated with the acquisition of a scan showing brain deformation. The sparse displacement field between the floating and reference images is estimated with the aid of volumetric block matching [9]. For each registration point, block matching iteratively searches for such a location of the region surrounding this point in the floating image (defined as search block), that maximizes the similarity metric with respect to the overlapping portion (search window) of reference image.

Depending on the properties of the intra-operative MRI (e.g., noise and modality), the choice of the best similarity metric to be used for block matching may not be clear, see Škerl et al. [28]. At the same time, in the current protocol of NRR there is usually significant time (≈30 min) between the acquisition of the first intra-operative scan and the MRI showing brain shift. This time can be used to perform intra-operative comparison of the similarity metrics for block matching. Škerl et alr̃eport that for their approach, *"the evaluations of similarity measures on a 3.2 GHz Pentium IV computer required about two months of CPU processing time"* [28]. Evidently, such computations can be enabled intra-operatively only with the aid of large scale computing resources.

The block matching result contains outlier displacements. The challenging questions in this phase are: how to remove the outliers, and how to approximate the brain deformation from a sparse and irregular set of displacements. A mesh model of the intracranial cavity is used to approximate the brain shift using FEM. This problem can be formulated as energy minimization. We seek the balance between the external forces, proportional to the sparse displacements, and the internal forces of the mesh resisting deformation:

$$[K + H^T S H]U = H^T S D. \tag{1}$$

K is the mesh stiffness matrix, H is the linear interpolation matrix form the matches to the displacements at mesh vertices, S is the block matching stiffness matrix (matches with higher confidence are assigned higher weights), D is the vector for the block displacements, and U is the unknown displacement vector for mesh vertices. The stiffness matrix K is calculated based on the assumed physical properties of the brain tissue E and ν (see Table 1). This formulation can tolerate outliers, but suffers from a systematic error with respect to the correctly estimated displacements. Alternatively, we can use approximation to compute the locations of vertices which minimize error with respect to the block-matches:

$$arg \min_U (HU - D)^T S (HU - D). \tag{2}$$

However, this formulation would also minimize displacement error w.r.t. outlier measurements, which we would like to eliminate from the set of block displacements. A robust iterative approach combines approximation and interpolation [9]. Gradual convergence to the interpolation solution is achieved through the use of the external force F added to the formulation (1) to slowly relax the internal mesh stress:

$$[K + H^T S H]U = H^T S D + F. \tag{3}$$

Table 1: Parameter space for the NRR approach.

parameter	default setting
Young modulus (E)	694 Pa
Poisson's ratio (v)	0.45
selected fraction of registration points (f_B)	10%
block connectivity ($bConn$)	26
search block dimensions ($bSize$)	$7 \times 7 \times 7$
search window dimensions ($wSize$)	$11 \times 11 \times 25$
search step ($sStep$)	$1 \times 1 \times 1$
number of rejection steps (n_R)	10
rejected blocks fraction (f_R)	25%
energy trade-off (α)	$\frac{trace(K)}{n}$
error model breakup point (λ)	0.5
CG precision (r_{CG})	0.001

Rejection of the outlier matches is done iteratively, with user-defined total percentage of matches to be discarded, f_R, and the number of rejection iterations, n_R, as follows:

INPUT: n_R, f_R
for $i = 0$ to n_R **do**
  $F_i \Leftarrow KU_i$
  $U_{i+1} \Leftarrow [K + H^T S H]^{-1}[H^T S D + F_i]$
  for all blocks k **do**
    compute error function ξ_k
  end for
  reject $\frac{f_R}{n_R}$ blocks with highest error function ξ
  recompute S, H, D
end for
repeat
  $F_i \Leftarrow KU_i$
  $U_{i+1} \Leftarrow [K + H^T S H]^{-1}[H^T S D + F_i]$
until convergence

The force F is computed at each iteration to balance the internal force of the mesh KU_i. The error ξ_k measures the difference between the block displacement approximated from the current deformed mesh and the matching target for the kth block. The user-defined percentage of the displacements with the highest ξ_k values are rejected. This method converges to the formulation in (2), and at the same time is tolerant to outliers.

NRR parameter space and accuracy assessment Registration accuracy, robustness, and performance can all be affected by the values of NRR parameters, which are summarized in Table 1. The optimum values of all these parameters are difficult to identify. For example, there is no consensus about the true values for the physical properties of the live tissue in the biomechanics community [10]. Optimum block and window sizes used during block matching depend on the properties of the images and the scale of brain shift. The optimum similarity metric to be used during block matching is also not always known [28].

Table 2: Time (minutes) required for sequential block matching with 100K registration points using Normalized Cross Correlation similarity metric.

	search window dimension							
block dimension	7	9	11	13	15	17	19	21
5	0.4	0.8	1.5	2.7	3.6	5.2	7.3	9.6
7	0.9	2.0	3.6	5.9	8.9	13.1	17.9	23.9
9	2.6	5.6	9.9	16.1	24.5	35.3	48.9	65.2
11	4.5	9.6	17.2	28.2	42.9	62.0	85.9	113.5

One approach to address the problem of sub-optimal parameter selection is to use *speculative execution* of the registration, as suggested by Ino et al. [19]. The idea of speculative execution is to compute multiple registrations on the same input data using different parameter settings. Variability between the results computed in this way allows to estimate sensitivity of the method, e.g., by automatically calculating certain metrics to assure registration accuracy. A number of such metrics for NRR have been proposed by Christensen et al. [8]. Archip et al. [4] used Hausdorff distance between edges identified automatically to prospectively evaluate the accuracy of the registration achieved by the algorithm of Clatz et al. [9]. Fedorov et al. [13] proposed a robust modification of the HD metric, which allows local evaluation of the registration accuracy in a region of the image. Automated approaches for NRR assessment are particularly useful in the context of the speculative execution.

The practical problem of performing speculative execution is obviously the enormous amount of computation required. For example, summing together the numbers in Table 2, we need more than 11 hours of computation on a high-end workstation (the timings were collected for the highly optimized code on Intel Xeon 3.7GHz), to sequentially complete the parameter study for block matching only. Considering that there may be 3-4 different similarity metrics that must be evaluated, the range of valid values for the outlier rejection in the solver, and the need to assess the accuracy of each NRR result, the total time required to perform exhaustive evaluation on a *single dataset* is in the order of days of sequential processing. Significantly more complex models of brain deformation have been presented to date. Dumpuri et al. [12] describe a predictive brain deformation model, that requires about 2 min to obtain the solution in parallel using 16 computing nodes. Considering all these factors, performing speculative studies on hundreds of datasets quickly becomes not practical even if a small computing cluster is available.

Grid resources have two important applications for NRR and IGNS. First, we need to complete (1) time-critical intra-operative computation of the NRR with the default parameter settings, and (2) intra-operative speculative execution. Using the grid-enabled NRR presented here we can also perform comparative evaluation of the similarity measures between the acquisition of the first intra-operative MRI, and the first MRI showing brain shift. In the current NRR protocol, computational resources are idle during this time period, and the single pre-defined similarity metric is used during block-matching. Second, distributed resources can be used to facilitate large-scale retrospective evaluations and parametric studies of the method on archived MRI brain shift data.

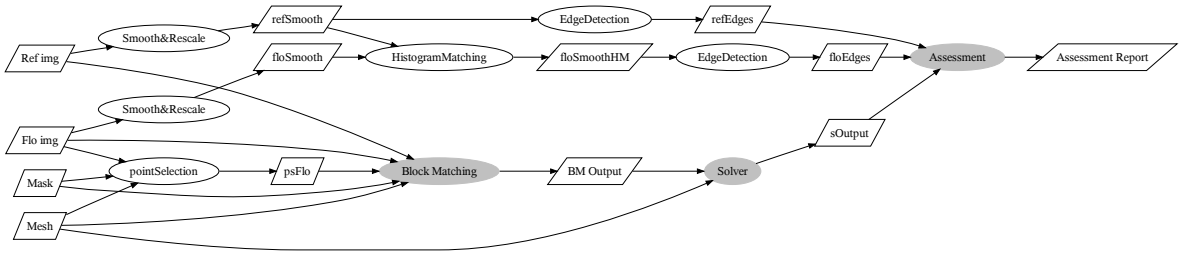

Figure 1: NRR workflow diagram for single registration execution (shaded are the time-critical components of the workflow).

4 Grid-enabled NRR workflow

We used the Swift workflow management system [30] for the implementation of the NRR workflow. Our motivation for using Swift over other similar systems were in particular the flexibility of its scripting language, transparent handling of the data distribution, and the minimum effort required to interface legacy applications from within SwiftScript. As we showed in [7], fault tolerance is of particular importance for the NRR execution. The Swift execution engine is capable of handling faults (e.g., those caused by intermittent resource unavailability) at the task level.

We have restructured the implementation of the NRR procedure, so that each processing step described in the previous Section is implemented as a separate C++ module. In order to launch an executable from the Swift workflow, the corresponding code must be compiled at each of the sites. The peculiarities of running the task are provided to Swift in the *transformation catalog*. The transformation catalog contains the identifier of the site where the executable is installed, together with optional information on its invocation. For example, in order to execute an MPI-parallelized task, the number of nodes must be explicitly defined. In order to guarantee that the task execution will be able to complete on the systems managed by a site-specific Local Resource Manager (LRM, e.g., PBS or LSF), estimated wall clock time must also be specified. All the TeraGrid sites are managed by some form of LRM.

The execution sites are described in the Swift *site catalog*. The site catalog contains a *pool element* for each execution site. The pool element specifies the file transfer method and task invocation method for the given site. In our case, for the local resource, the file transfer method is local filesystem copy, and task invocation is local execution. For the pool elements corresponding to the TeraGrid sites, file transfer is done through GridFTP (each TeraGrid site provides a GridFTP service URL for file operations), and the remote resources are accessed via the Globus GRAM4 [15] gatekeeper endpoint. Although the complexity of task submission is significant while running jobs on the remote resources as compared to the local ones, it is handled transparently by the Swift execution engine and CoG Karajan.

Single NRR workflow The NRR workflow diagram of a single NRR procedure together with the assessment module is shown in Figure 1. The block matching task is parallelized using MPI, and has been deployed on the TeraGrid sites for remote parallel execution. The other components of the workflow are executed on the local resources (single node of the W&M SciClone cluster). This NRR workflow corresponds to the base case for computation

Table 3: Summary of environments and processing times on a single node for the time critical components of the NRR workflow: sequential block matching (C1), solver (C2), registration assessment (C3). 55K registration points, $bSize = 7$, $wSize = 17$, ≈4.5K mesh DoF.

	W&M SciClone	TACC Lonestar	NCSA Abe	NCSA Mercury
	Opteron 2218 2.6GHz, 8GB	Intel Xeon 2.6GHz, 8GB	Intel 64 Clowertown 2.33GHz, 8GB	Intel Itanium 2 1.5GHz, 4GB
	gcc 4.1.2, -O3 MVAPICH2-1.0.1	icc 10.1, -O3 MVAPICH-1.0.1	gcc 3.4.6, -O3 MPICH-VMI 2.2.0-3	gcc 3.2.2, -O3 MPICH-GM 1.2.5
C1	275 sec	294 sec	418 sec	608 sec
C2	6 sec	27 sec	12 sec	21 sec
C3	24 sec	22 sec	182 sec	495 sec

using our custom implementation in [7], augmented with the accuracy assessment module.

Speculative NRR workflow With SwiftScript, we need just a few additional lines to iterate through the user-defined parameter space. Using the Swift filesystem mappers, each intermediate and the end result of the speculative computation is given a unique self-descriptive name, and stored in a user-defined location on the local filesystem. For the speculative NRR workflow, block matching, solver, and NRR assessment tasks are executed on the remote TeraGrid resources. The rest of the tasks are computed locally. The complete diagram of the speculative execution of the NRR workflow is shown in Figure 2.

In summary, we see the Swift-based workflow description of NRR as a significant improvement over the custom implementation in [7]. In this new implementation we are able to easily include additional computing resources for NRR execution. Modification of the procedure to include new intermediate steps, or even major changes in the computation (like speculative execution) requires minimum updates to the existing workflow description.

5 Results

We used three sites of TeraGrid (TACC Lonestar, NCSA Abe, NCSA Mercury) to install components of the workflow that we selected for remote execution. These sites were chosen

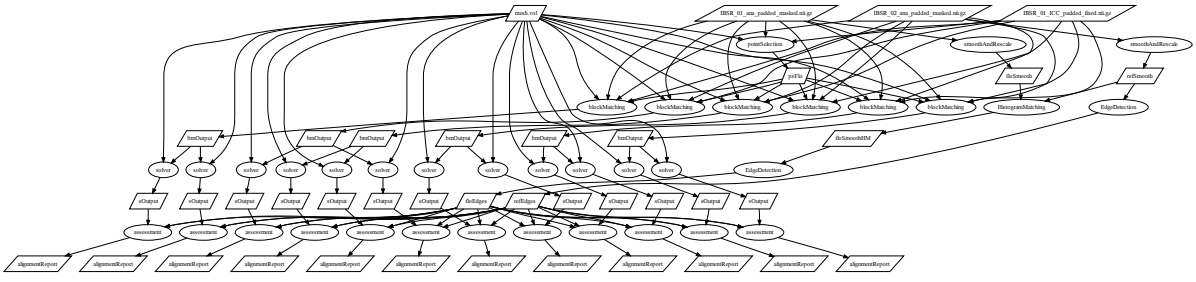

Figure 2: NRR workflow diagram for speculative execution of the registration with $bSize \in \{7, 9\}$, $wSize \in \{15, 17, 19\}$, and $f_R \in \{0.2, 0.3\}$.

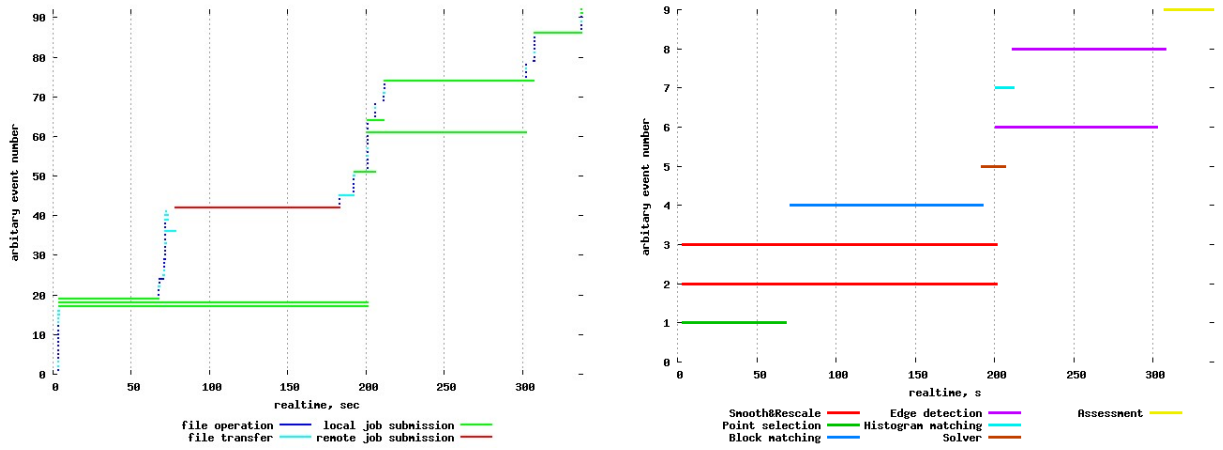

Figure 3: Execution timeline for single instance of NRR on NCSA Abe site. Left: file operations and task execution summary. Right: task executions colored by task type (file operation times are included).

among the currently available 11 TeraGrid sites because they provide the most convenient development environment (multi-core Intel nodes). Our implementation of the NRR has been originally developed and tuned in similar environments. We expected that porting of the codes would require less work for the selected sites as compared to, for example, BlueGene/L or Cray XT3 platforms, available within TeraGrid.

Installation of the NRR code included compilation of the supporting open source libraries at the TeraGrid sites we use. As part of the system setup, we configured the local firewall to allow incoming connections from the TeraGrid sites (this is a necessary condition for using remote Globus GRAM service). Access to the TeraGrid resources is simplified with a single sign-on procedure and the Grid Security Infrastructure [15]. The user password is entered once at the local site (SciClone), and after that all job submissions and data transfers to and from the TeraGrid sites are transparent from the SciClone user perspective.

The configurations for the selected sites is summarized in Table 3. Based on this summary, there are two observations. First, hardware configurations were introduced at different times on different TeraGrid sites. Therefore, there is significant heterogeneity and disparity in processing power. Second, it is not possible to use the same compiler and MPI implementation on all TeraGrid sites. This may create significant difficulties for porting legacy codes, since different compilers may require different configurations and, sometimes, may simply fail to compile the code. As the results of the benchmark runs for the installed components show (see Table 3), performance vary significantly for the same level optimization but different compilers, amplified even more by the non-uniformity of clock speeds.

Single NRR instance execution We performed experimental evaluation of the single instance of NRR workflow (see Figure 1) at each of the three sites of the TeraGrid in Table 3. In each case, block matching was executed on 10 nodes at the remote site, while the rest of the workflow components were run on the single node of the SciClone cluster at W&M. The execution timeline for NCSA Abe cluster is summarized in Figure 3. Although we did not use any advance resource reservation capabilities, which are available at some sites of TeraGrid, we were able to complete execution and collect traces without major queuing delays at

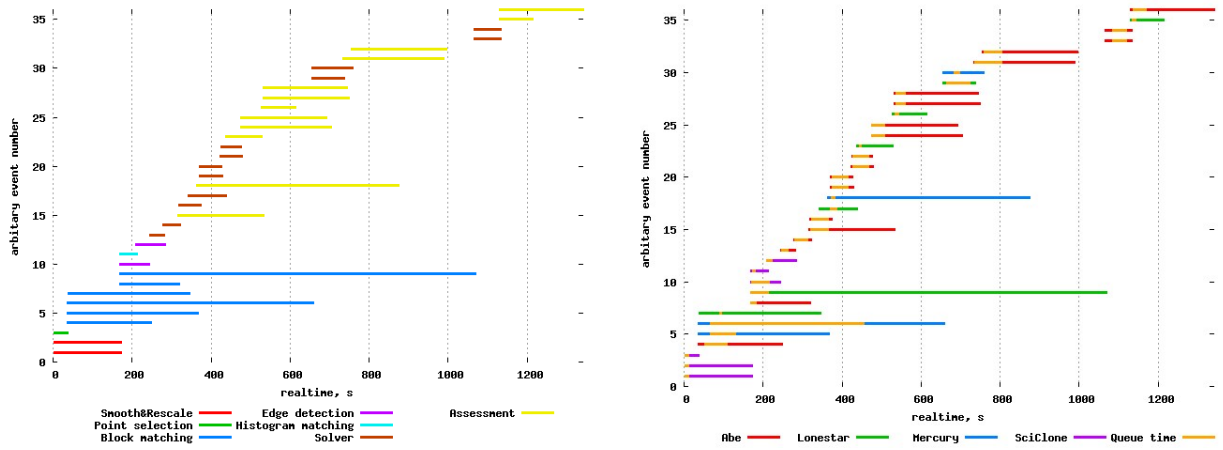

Figure 4: Speculative NRR timeline for the parameter space in Figure 2. Left: task executions colored by the execution site. Right: tasks executions colored by the task type (file operation times and queued times are included). *This figure is best viewed in color*

each of the tested sites. The file transfer delays (maximum below 30 sec) are negligible in comparison to the speed improvement in block matching computation due to parallelization in each case. This is in part attributed to high-performance GridFTP protocol, which in our experience provides faster transfer of image data as compared to SFTP that we used in [7]. The NRR workflow was completed in less than 6 min (this includes point selection, which can be evaluated before the time-critical part of the computation).

Speculative NRR execution The timeline of speculative NRR execution for a relatively small parameter search space is shown in Figure 4 (this corresponds to the workflow diagram in Figure 2). The total time required to complete such speculative execution was in this particular case about 20 min. Observe, that the execution of the single block matching component on Lonestar site reportedly took more than 800 sec in the speculative run. We attribute this to the intermittent overload of the remote GRAM service, which might have prevented the timely propagation of the job update status to the submitting client.

The resources available within the TeraGrid are sufficient for concurrent execution of all independent branches of the workflow shown in Figure 2. Therefore, in the ideal case, ignoring all overheads, one could expect that total time required for speculative execution would be equal to the longest single execution of registration. We measured this time by executing a single instance of NRR on each of the sites, and the maximum execution time we observed was under 10 min (without accounting for the queue delays). In practice, a number of factors contribute to the increased processing time, as compared to the ideal case:

(1) As we show in Table 3, time required to complete the same task varies depending on the selected TeraGrid site. The current procedure of the workflow scheduling by Swift execution engine does not take into account execution time during the process of resource selection (the same task is available for execution at all three sites).

(2) The resources at each site are available through the remote LRM, therefore there are queue delays. In our case, as shown in Figure 4, cumulative queuing delays for all jobs were about 20 minutes (!), with the maximum individual queued time of 385 sec.

(3) There are constraints in the Swift scheduler that limit the maximum number of jobs

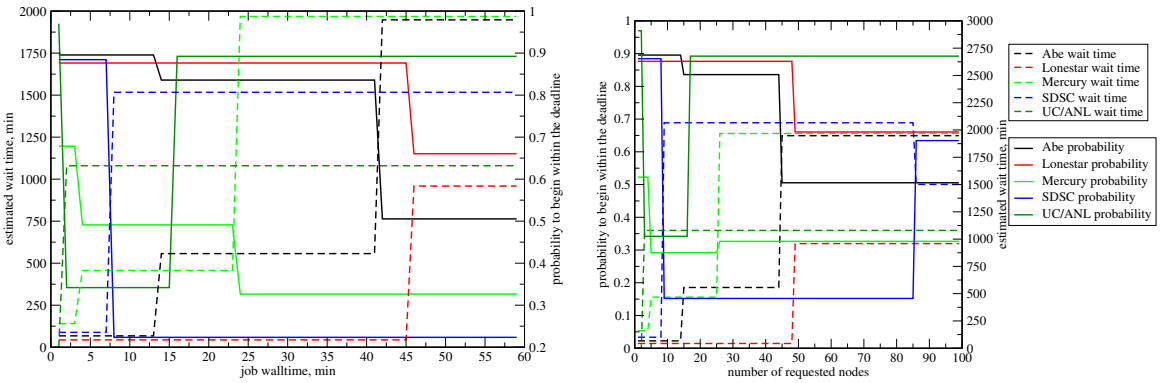

Figure 5: Estimation of queue wait times and probability that the estimate is correct by QBETS service [23] for some of the TeraGrid sites (job start within 30 min with requested 95% confidence in the result). Left: estimation as a function of job walltime, 16 nodes requested. Right: estimation as a function of the number of nodes, 30 min job wall time.

allowed to run at a given site at the same time (the *job throttle* limit). This explains why there are only 4, not 6, concurrent executions of block matching initially, see Figure 4. The constraints are in place to prevent overloading the GRAM service with the large number of concurrent submissions.

(4) There are multiple minor overheads due to file staging (transfers of input and output files), task scheduling, Swift book-keeping, etc. Current scheduling policy used in Swift does not account for minimization of the file transfers (e.g., in Figure 4, observe that assessment task is performed on sites different from where solver was run to produce registration results). These are, however, minor issues compared to the first three items.

The major problem is how to reduce the impact of the queue delays at the remote site. One approach, which has been considered by the community, is to use batch queue wait time prediction techniques to adjust the configuration of the job (e.g., number of nodes and walltime) for better throughput. Queue prediction services are currently provided by all the TeraGrid sites, and we show how job configuration can reportedly affect the wait time. The dynamic selection of the job configuration can potentially decrease queue wait time.

A conceptually different approach is to use multi-level scheduling. In this case, a group of nodes within a cluster is allocated, and then used to schedule multiple smaller tasks of the workflow. Falkon [25] implements this concept, and has been shown to achieve very high throughput for applications they evaluated. The use of multi-level scheduling also helps to alleviate the restriction on the task concurrency, as scheduling of the smaller jobs within the larger allocation would not require interaction with GRAM. We expect to see significant benefits in using Falkon and other similar techniques under development (e.g., the *Coasters* resource provider within Swift) for executing speculative NRR workflows.

File transfer costs were insignificant for the reduced parametric studies we performed. However, for the full-scale parametric studies workflow scheduling will need to take into account data affinity to minimize communication. Also, since the execution time of the solver module is comparable with the time required to transfer its input, it is very inefficient to schedule solver as a separate task. Again, multi-level scheduling and job clustering will implicitly improve execution locality and minimize file transfers.

6 Discussion

As a result of this work, we developed a workflow-based implementation of the single and speculative execution of the NRR method, and completed its preliminary evaluation at three TeraGrid sites. We suggest a number of modifications to the current IGNS registration protocol, that leverage high-performance computing resources, and can potentially improve the accuracy of registration. The implementation we developed is not tied to a specific collection of execution sites, as the framework presented in [7], but is generic enough due to its use of Swift, and the broad range of resource providers supported by Swift (e.g., Globus GRAM2 and GRAM4, PBS and LSF batch schedulers, etc.). Therefore, the implementation can be deployed with minimum effort on any conventional cluster within a hospital, or can be executed on the TeraGrid resources, to enable large-scale population and parameter studies of the NRR method.

The important result of our evaluation is that it is feasible to use TeraGrid resources for intra-operative NRR. Although we were able to complete test runs without significant queue delays for the single NRR instance, advance reservations should be used, whenever possible, to guarantee resource availability during the surgery. Our preliminary results of evaluating speculative NRR workflow executions indicate, that with the current technology we cannot achieve the required task throughput to do speculative execution within the time constraints of IGNS. However, the work under development in the relevant projects, e.g., Falkon [25], is very promising. We expect to achieve significant performance improvements by using multi-level scheduling techniques, as compared to the conventional direct interaction with the remote batch scheduler via Globus GRAM4 for each workflow task. Again, due to our use of Swift for workflow implementation, we can leverage these new execution models without any modifications to the NRR code or workflow script.

The open question that cannot be answered by this study (yet) is this: Considering the availability of the local mid-size computing cluster at an institution, is the effort to migrate an application like NRR on the TeraGrid justified? As we discovered during the course of this work, significant time investment may be required to port a code on the TeraGrid and learn the site-specific usage procedures. Efficient use of the TeraGrid resources is complicated by the queue delays, scheduled and emergency shutdowns of the TeraGrid sites and Globus services we use, and by the other detailed practical problems we list in Section 5. On the other hand, although mid-size clusters (100-200 nodes) become more and more ubiquitous, we believe they are not yet affordable by an average hospital or research group. We also believe that local clusters (which by the way are usually also managed by a batch scheduler) are insufficient for large-scale computations like NRR speculative execution, or large-scale population studies. It remains to be seen how practical the full-scale speculative studies can be on the TeraGrid. However, the preliminary results we present are sufficient to affirm the advantages of the Swift-based workflow NRR implementation, which can be used immediately both on TeraGrid and conventional clusters for NRR evaluation. At the same time, to achieve good performance of the speculative NRR execution we need to evaluate non-conventional, throughput-oriented workflow scheduling techniques.

Our future work will be focused on the design and evaluation of the new scheduling and execution methods in the framework of the Swift workflow engine, with the goal of improving performance of the speculative NRR execution. We will also use the presented

implementation for large scale evaluation of the retrospective brain shift MRI data. Close integration of the workflow specification and execution with the available end-user tools for medical image analysis[1] (or, alternatively, implementation of a Science gateway interface) will be essential to make this technology useful for the medical image research community.

References

[1] Biomedical Informatics Research Network, 2008. `http://www.nbirn.net`.

[2] Grid-Enabled Medical Simulation Services, 2008. `http://www.it.neclab.eu/gemss/`.

[3] MammoGrid, 2008. `http://www.cems.uwe.ac.uk/cccs/project.php?name=mammogrid`.

[4] Archip, N., Clatz, O., Whalen, S., Kacher, D., Fedorov, A., Kot, A., Chrisochoides, N., Jolesz, F., Golby, A., Black, P. M., and Warfield, S. K. Non-rigid alignment of pre-operative MRI, fMRI, and DT-MRI with intra-operative MRI for enhanced visualization and navigation in image-guided neurosurgery. *Neuroimage, 35*, 2 (4/1 2007), 609–624.

[5] Beckman, P., Nadella, S., Trebon, N., and Beschastnikh, I. SPRUCE: A system for supporting urgent high-performance computing. In *Proc. of WoCo9: Grid-based Problem Solving Environments* (2006).

[6] Catlett, J. et al. TeraGrid: Analysis of organization, system architecture, and middleware enabling new types of applications. *Advances in Parallel Computing 16* (2008), 225–249.

[7] Chrisochoides, N., Fedorov, A., Kot, A., Archip, N., Clatz, O., Kikinis, R., and Warfield, S. K. Toward real-time image guided neurosurgery using distributed and grid computing. In *Proc. of the 2006 ACM/IEEE Conference on Supercomputing* (2006).

[8] Christensen, G. E., Geng, X., Kuhl, J. G., Bruss, J., Grabowski, T. J., Pirwani, I. A., Vannier, M. W., Allen, J. S., and Damasio, H. Introduction to the non-rigid image registration evaluation project (NIREP). In *Proc. of WBIR 2006* (2006), pp. 128–135.

[9] Clatz, O., Delingette, H., Talos, I. F., Golby, A. J., Kikinis, R., Jolesz, F. A., Ayache, N., and Warfield, S. K. Robust non-rigid registration to capture brain shift from intra-operative MRI. *IEEE Transactions on Medical Imaging 24*, 11 (2005), 1417–1427.

[10] Delingette, H., and Ayache, N. *Soft tissue modeling for surgery simulation*, 1 ed., vol. XII of *Handbook of Numerical Analysis: Special volume: Computational models for the human body*. Elsevier, Netherlands, 2004, pp. 453–550.

[11] Dong, S., Insley, J., Karonis, N. T., Papka, M. E., Binns, J., and Karniadakis, G. Simulating and visualizing the human arterial system on the TeraGrid. *Future Generation Computer Systems, 22*, 8 (10 2006), 1011–1017.

[12] Dumpuri, P., Thompson, R. C., Dawant, B. M., Cao, A., and Miga, M. I. An atlas-based method to compensate for brain shift: Preliminary results. *Medical Image Analysis, 11*, 2 (4 2007), 128–145.

[13] Fedorov, A., Billet, E., Prastawa, M., Radmanesh, A., Gerig, G., Kikinis, R., Warfield, S. K., and Chrisochoides, N. Evaluation of brain MRI alignment with the robust hausdorff distance measures. In *Proc. of ISVC 2008 (to appear)* (2008).

[1] In particular, 3DSlicer, `http://www.slicer.org`, which is widely used by the medical researchers at Brigham and Women's Hospital.

[14] Ferrant, M., Nabavi, A., Macq, B. M., Jolesz, F. A., Kikinis, R., and Warfield, S. K. Registration of 3d intraoperative MR images of the brain using a finite element biomechanical model. *IEEE Transactions on Medical Imaging 20*, 12 (2001), 1384–1397.

[15] Foster, I. Globus toolkit version 4: Software for service-oriented systems. *Journal of Computer Science and Technology 21*, 4 (07/28 2006), 513–520.

[16] Foster, I., and Kesselman, C. *The Grid: A blueprint for a new computing infrastructure*. Morgan Kaufmann, San-Francisco, 1998.

[17] Glatard, T., Pennec, X., and Montagnat, J. Performance evaluation of grid-enabled registration algorithms using bronze-standards. In *Proc. of MICCAI 2006* (2006), pp. 152–160.

[18] Gropp, W., Haber, E., Heldmann, S., Keyes, D., Miller, N., Schopf, J., and Yang, T. *Grid-based Image Registration*, vol. 239 of *Grid-Based Problem Solving Environments*. Springer, 2007, pp. 435–448.

[19] Ino, F., Kawasaki, Y., Tashiro, T., Nakajima, Y., Sato, Y., Tamura, S., and Hagihara, K. A parallel implementation of 2-d/3-d image registration for computer-assisted surgery. *Int J for Bioinformatics Research and Applications 2*, 4 (2006), 341–357.

[20] Lippmann, H., and Kruggel, F. Quasi-real-time neurosurgery support by MRI processing via grid computing. *Neurosurgery Clinics of North America 16*, 1 (2005), 65–75.

[21] Majumdar, A., Birnbaum, A., Choi, D. J., Trivedi, A., Warfield, S. K., Baldridge, K., and Krysl, P. A dynamic data driven grid system for intra-operative image guided neurosurgery. In *Proc. of ICCS 2005* (2005), pp. 672–679.

[22] Manos, S., Zasada, S., Mazzeo, M. D., Haines, R., Doctors, G., Brew, S., Pinning, R., Brooke, J., and Coveney, P. V. Patient specific whole cerebral blood flow simulation: A future role in surgical treatment for neurovascular pathologies. In *Proc. of Teragrid'08* (2008).

[23] Nurmi, D., Brevik, J., and Wolski, R. QBETS: Queue bounds estimation from time series. In *Proc. of JSSPP'07* (2008), pp. 76–101.

[24] Parashar, M., and Lee, C. A. Special issue on grid computing. *Proceedings of IEEE 93*, 3 (2005), 479–483.

[25] Raicu, I., Zhang, Z., Wilde, M., Foster, I., Beckman, P., Iskra, K., and Clifford, B. Toward loosely coupled programming on petascale systems. In *Proc. of Supercomputing 2008 (to appear)* (2008).

[26] Stefanescu, R., Pennec, X., and Ayache, N. Grid powered nonlinear image registration with locally adaptive regularization. *Medical Image Analysis 8*, 3 (9 2004), 325–342.

[27] von Laszewski, G., Hategan, M., and Kodeboyina, D. *Java CoG kit workflows*. Workflows for eScience. Springer, 2007, pp. 340–356.

[28] Škerl, D., Likar, B., and Pernuš, F. A protocol for evaluation of similarity measures for non-rigid registration. *Medical Image Analysis, 12*, 1 (2008), 42–54.

[29] Warfield, S. K., Jolesz, F. A., and Kikinis, R. A high performance computing approach to the registration of medical imaging data. *Parallel Computing, 24*, 9-10 (9 1998), 1345–1368.

[30] Zhao, Y., Hategan, M., Clifford, B., Foster, I., von Laszewski, G., Nefedova, V., Raicu, I., Stef-Praun, T., and Wilde, M. Swift: fast, reliable, loosely coupled parallel computation. In *Proc. of 2007 IEEE Congress on Services* (2007), pp. 199–206.