Enabling Parallel Scientific Applications with Workflow Tools

Adam Lathers, Mei-Hui Su, Alex Kulungowski, Abel W. Lin, Gaurang Mehta, Steven T. Peltier, Ewa Deelman, and Mark H. Ellisman

***Abstract*--Electron tomography is a powerful tool for deriving three-dimensional (3D) structural information about biological systems within the spatial scale spanning 1 nm^3 and 10 mm^3. With this technique, it is possible to derive detailed models of sub-cellular components such as organelles and synaptic complexes and to resolve the 3D distribution of their protein constituents in situ. Due in part to exponentially growing raw data-sizes, there continues to be a need for the increased integration of High-Performance Computing (HPC) and Grid technologies with traditional electron tomography processes to provide faster data processing throughput. This is increasingly relevant because emerging mathematical algorithms that provide better data fidelity are more computationally intensive for larger raw data sizes. Progress has been made towards the transparent use of HPC and Grid tools for launching scientific applications without passing on the necessary administrative overhead and complexity (resource administration, authentication, scheduling, data delivery) to the non-computer scientist end-user. There is still a need, however, to simplify the use of these tools for applications developers who are developing novel algorithms for computation. Here we describe the architecture of the Telescience Project (http://telescience.ucsd.edu), specifically the use of layered workflow technologies to parallelize and execute scientific codes across a distributed and heterogeneous computational resource pool (including resources from the TeraGrid and OptIPuter projects) without the need for the application developer to understand the intricacies of the Grid.**

***Index Terms*--Grid Computing, Pegasus, Telescience, Workflow**

I. Introduction

More than a decade ago researchers at the National Center for Microscopy and Imaging Research (NCMIR) demonstrated the feasibility of remote control of bio-imaging instruments (SIGraph Conference 1992). The progression of that nascent software system was achieved under an NSF Grand Challenge Award for the Collaboratory for Microscopic Digital Anatomy (CMDA) that delivered the first production Telemicroscopy™ [1][2] software system, released in 1999. Over the last 5+ years, researchers at NCMIR have developed an end-to-end system known as the Telescience™ Project [3][4][5] that combines the use of Telemicroscopy with tools for parallel distributed computation, distributed data management and archival, and interactive integrated visualization tools within a single sign-on portal. Using 2D and 3D multi-scale imaging as the scientific driver, this system brings to bear various HPC and Grid components (that were previously developed in isolation) to the scientific process.

While Telescience (and other Grid projects such as BIRN [6] and GEON [7]) have had increasing success in delivering HPC and Grid functionality to the end-user, little progress has been made in simplifying adoption of these capabilities by the domain specific applications developer. Often there exists an impasse during the integration of scientific codes with the Grid. Application developers are required to become Grid experts or Grid developers are required to gain expertise in the application domain.

The contributions of this paper are the development of a novel computational infrastructure that utilizes workflow technologies to accelerate the time-to-solution for creating Grid-based scientific algorithms. Here we describe the use of this infrastructure to create a parallel, Grid-based application from its origins in a MATLAB development environment. We also discuss how the workflow technologies can be integrated within user-friendly portal-based environments to deliver high-performance, Grid-based computations to non-Grid experts.

II. Scientific Motivation

Researchers at NCMIR are leaders in the area of electron microscopic tomography. In this cutting edge tomography, there are three main stages involved in generating 3D volumes from 2D transmission electron microscope (TEM) projection data. The first stage, feature tracking, is a laborious process of marking fiducial points (usually nanometer sized colloidal gold) throughout a tomographic tilt series and utilizing reprojection errors to fine tune the correspondences. Automated and semi-automated methods to track features in TEM projection series do exist, but their performance, especially for large tilt series (>$4K^2$ pixels in XY), is often suboptimal. Thus, while there continue to be efforts to automate the process, due to the frequent user intervention required to achieve the necessary level of

Manuscript received March 20, 2006. This work was supported in part by grants from the National Institutes of Health (NINDS NS046068, P41 RR004050, and P41 RR008605 NBCR) and the National Science Foundation (ANI0225642)

A. Lathers, A. Kulungowski, A.W. Lin, S.T. Peltier, and M.H. Ellisman are with the National Center for Microscopy and Imaging Research, University of California at San Diego, La Jolla, CA 92093-0608, USA (Corresponding author email: awlin@ncmir.ucsd.edu)

M. Su, G. Metha, and E. Deelman are with the Information Sciences Institute, University of Southern California Marina Del Rey, CA 90292, USA

refinement in the correspondences, this step is currently not suitable for Grid-based parallelization.

The second stage is composed of three individual parts: (a) the calculation of the final alignment transformations using the correspondences established in the first stage, (b) the transformation of the projections, and (c) the backprojection setup. Typically, parts (a) and (c) do not consume an inordinate amount of time; however, computing the transformed projections (part b) can be quite computationally intensive (relative to the number of transformations involved). The third (and final) stage is the actual backprojection, which produces the reconstructed 3D volume from the transformed projection series. This is where the vast majority of time and processing power is spent. Due to the amount of computation required (several weeks on a single workstation) and the embarrassingly parallel structure of the underlying process, the second and third stages are well suited for parallelization and Grid computing paradigms.

A challenge faced by research organizations such as NCMIR (and generalizeable to other organizations) is whether to allocate limited human resources towards developing parallel, Grid-based codes or to spend those resources refining these advanced algorithms. This quandary is amplified by a lack of mature technologies to reduce the threshold for applications developers to build parallel Grid enabled tools. NCMIR developers have created the Telescience ATOMIC [8] toolkit to simplify the process of integrating gross Grid capabilities with applications (job launching, security, etc), but little exists to help navigate the requirements associated with coordinating these capabilities within the context of a parallel application.

III. Telescience Architecture and Workflow Hierarchy

The Telescience architecture (shown in Figure 1) has a hierarchal design, reducing the complexity of each layer relative to its immediate neighbors, allowing developers of each layer to focus on the "business logic" and to have the "presentation logic" be managed by the layer above. For example a portlet developer needs only be concerned with the core business logic of the portlet, the presentation of that portlet's functionality is managed by the higher-level presentation services. Likewise, the portlet developer utilizes the ATOMIC presentation API and need not be concerned with the business logic of the underlying services.

Primary user interaction within the Telescience Project occurs via a web portal interface. While the Telescience Project currently utilizes the GridSphere Framework, (http://www.gridsphere.org) [9] any JSR168 compliant portal framework [10], such as JetSpeed [11] or even a desktop application can be the primary user interface within the Telescience architecture.

Even within an architecture aimed at reducing the complexity of software interactions, there is still a temptation to build all-encompassing applications that capture all the necessary functionality (across layers) in a single program. Workflow tools accelerate the rate of software creation while reducing this tendency by working across layers to link together disparate code fragments and/or applications (some pre-existing) into a single virtual environment, with little to no change to the original source code. Within the context of the Telescience architecture and computing environment, we believe that workflow tools fall into the following hierarchical classes:

- Process Management Workflows that frame the highest level scientific (laboratory) process and provide policy, process, state management, and administrative tools including the coordination and management of lower level workflows/pipelines that may comprise a scientific study (or instance within that study);
- Inter-application Workflows that bridge together existing tools to streamline computational operations; and provide mechanisms to build and replicate computational processes
- Intra-application Workflows that are composed of planners and execution engines that optimize the execution of these plans on heterogeneous physical resources.

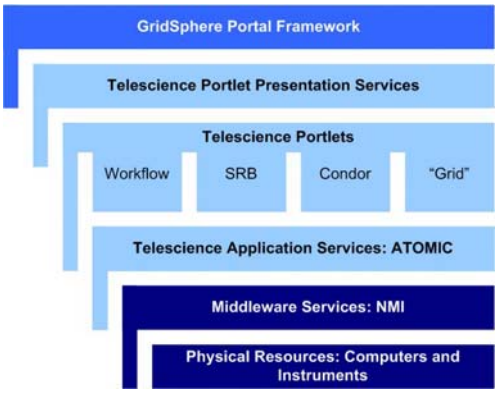

Figure 1: The Telescience architecture insulates users and developers from the complexities of the middleware infrastructure, linking client side resources to distributed physical resources. The Telescience project focuses on the interaction of systems software between the portal and the core middleware services (2nd to 4th boxes from the top).

The Telescience architecture facilitates the coordination and sharing of state information among these three workflow layers. Each layer has unique abilities and requirements. Process and state management tools (typically portal-based) are necessary to preserve and delegate the contextual information with regard to the user. This information includes process management, authentication and authorization, and high-level state information (represented as the workflow portlet in Figure 1). Inter-application tools create process pipelines, which are subcomponents of the highest-level experimental process management workflow. These tools are typically user driven GUI environments that are either ordered within the process management workflow or presented as a general tool to serve the process management workflow as needed. The lowest level "intra-application" tools are composed of sub-components of the "inter-application" tools and are necessary to map heterogeneously parallel tools to a heterogeneous pool of physical resources.

In this article we focus on the use of the intra-application class workflow tools to bring scientific algorithms to the Grid

faster than could otherwise be accomplished without these tools. In particular, we focus on the use of the Pegasus planner to map scientific workflows to the Grid, and ultimately the use of Condor DAGMan [12] to execute the workflows. Pegasus [13][14][15][16], which stands for Planning for Execution in Grids, is a framework that maps complex scientific workflows onto distributed resources, such as the Grid. Pegasus maps an abstract workflow description to its executable form and Condor DAGMan executes the jobs specified in the executable workflow. Pegasus and DAGMan are able to map and execute workflows on a variety of platforms: Condor pools, clusters managed by LSF or PBS, TeraGrid hosts, and individual hosts.

Pegasus operates on abstract workflow descriptions where the analysis is described in terms of application components and the data that the components use. The workflow is abstract because it does not identify the resources necessary for execution. Pegasus takes this abstract workflow description and produces an executable workflow which identifies the compute resources needed and includes data management nodes which stage the data in and out of the computations. Additional workflow nodes are added to register the newly derived data products so that they can be located at a later time.

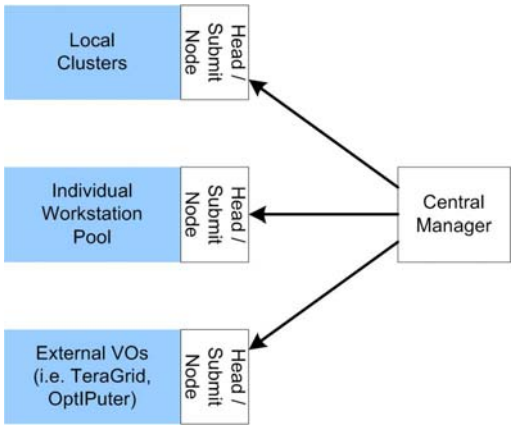

Figure 2: NCMIR's scientific computing environment places a common interface across a heterogeneous mix of resources

The first step in the process was to develop a unified scientific computing environment for the Telescience Project. Like many distributed virtual organizations, Telescience has access to a series of heterogeneous internal resources and is a scientific gateway for a number of external virtual organizations such as the TeraGrid (http://www.teragrid.org) and the OptIPuter (http://www.optiputer.net). With such a heterogeneous mix of resources, it was necessary to create an environment that had a common interface so that the workflow description would be easier to develop. Figure 2 outlines the scientific computing environment at NCMIR that is utilized by the Telescience Project.

A notable implementation choice of the Telescience scientific computing environment that differs from other distributed computing environments is that the Submit Node and the Central Manager do not share a file system with the worker nodes. This is in part to ensure modularity across the architecture, treating both internal and external pools of resources in the same manner. While this modularity provides a more consistent image for connecting to the worker resources, it has a number of implications for designing the system. The most significant is the need for robust systems for data delivery and authentication, authorization, accounting, and auditing (AAAA) services. Telescience is working with the TeraGrid project to refine the development AAAA services to effectively address the needs of application VOs like Telescience [17]. Implications of data transfers described in subsequent sections of this article.

IV. Scientific Driver: Electron Tomography

The software suite we have used in this experiment is the Transformed based Tracking, Bundle adjustment, and Reconstruction (TxBR) package [18] developed at NCMIR. TxBR relies on bundle adjustment to estimate simultaneously the 3D positions of the tracked feature points and the transforms of these positions associated with each projection in the series. Bundle adjustment is a useful technique for aligning TEM projection data because it accounts for the nonlinear distortions produced by the twisting of the electron trajectories in a TEM's magnetic lenses. The backprojection algorithm implemented in TxBR is an extension of the standard orthogonal backprojection process adapted to the general case of curvilinear electron trajectories [19]. While the transformation step of the second stage is a time--consuming process in its own right, as an initial test of the system we have currently only parallelized TxBR's backprojection (3rd stage).

There are three versions of TxBR's backprojection code: projective, quadratic, and cubic. As the names suggest, each version in the ordered list capitalizes upon a higher degree of approximation in the alignment transformations, albeit with a concomitant increase in processing time.

Quadratic and cubic backprojection both require significant amounts of processing time, with the serial runtime of a quadratic backprojection of a standard projection series measured in days, and that of a cubic backprojection of the same series measured in weeks. These long runtimes are partially the result of the experimental status of the TxBR codebase. With the exception of a C version of the quadratic backprojection program parallelized for clusters using MPI, all the programs and function libraries comprising TxBR are developed and implemented solely as MATLAB M-files. While the conversion of MATLAB M-files to C/C++ (or other compiled programming languages) may be undertaken and will ultimately yield more efficient run times (whether parallel Grid-enabled or not), the manpower effort required to undertake that conversion comes at the cost of refining the backprojection algorithm itself (as the application developer is is no longer refining the algorithm but rather spending time on programmatic language conversions).

To address this issue, Telescience capitalizes on the planning and execution capabilities of Pegasus and Condor to remove the requirement of the developer to code the logic of the parallelism directly into the application. That logic is ultimately captured within a Directed Acyclical Graph (DAG), allowing the developer to concentrate on the core process algorithm to be executed on the distributed resources. Figure

3 shows excerpts of sample DAG, submit, and component files that were manually created for this experiment. To further reduce the burden on applications developers, we have adopted the use of the Pegasus workflow planner.

```
DAG
[…]
Job Job1 1.submit
Job Job2 2.submit
Job Job3 3.submit
[…]
Submit File
Universe = vanilla
Executable = matlab tool run.sh
[…]
Arguments =
    run quadratic recon subset mlp ko3 1 1
[…]
Requirements = (FileSystemDomain =!= "")
   && (Arch =!= "IA64")
   && (Arch =!= "INTEL")
   && (Memory >= ImageSize)
   && ((OpSys == "LINUX")
   || (OpSys == "SOLARIS29")
   || (OpSys == "SOLARIS5.10") )
should_transfer_files = IF_NEEDED
transfer executable = False
when to transfer output = ON EXIT
Queue
matlab tool run.sh
[…]
basedir="M-Files"
executable="matlab -nodesktop -nosplash -r"
archID=`uname -m`
args="'$2', '$3', '$4'"
[…]
case $archID in
    x86 64)
    LD LIBRARY PATH="[…]/lib/glnxa64/"
    export LD LIBRARY PATH
esac
[…]
cd $basedir
hostname
exec nice $executable "$1($args)"
```

Figure 3: Example of DAGs, Submit Files, and other components

With this system, the developer supplies the necessary instructions to describe the logic underlying the parallelization using a single and simple abstract workflow in an XML format (DAX file). Pegasus then automatically produces an executable workflow from this DAX, identifying all necessary end data and computational resources that are needed (and eventually coordinating the use of resource discovery and scheduling tools to generate tuned workflows on demand).

As an initial trial we employed a embarrassingly parallel division of labor, assigning to each participating process the reconstruction of a contiguous subset of the entire reconstructed volume (along the Z axis). Using Pegasus and Condor, we were able to launch the MATLAB M-files on a number of heterogeneous systems within our lab. Using this method we have already achieved an approximate 6x increase in throughput, from 12+ days to just under 2 days for a "standard volume" (approximately 17GB raw data resulting in a 45GB 3D volume).

Currently this version is up and running for end-users from the Telescience Portal. Similar to the previous computational applications, the user interface completely shields the user from the components shown in Figure 3. Rather, the user is presented with a simplified GUI where only biologically relevant contextual information is required. The job is launched by a single button click and an email is sent to the user upon completion of the job.

SUN4u/SOLARIS29	4
SUN4u/SOLARIS5.10	10
X86_64/LINUX	8
Total	22

Figure 4: Resources Used During a Trial Run using Matlab M-Files (each machine averages 1GB RAM/ per processor across a 100MB-1GB network)

We believe that we can further increase the throughput for three primary reasons. First, our test currently only utilizes 22 of the CPUs within the internal NCMIR resources (see Figure 4); due to network performance issues and potential MATLAB licensing concerns, external resources such as TeraGrid were not included (see discussion for details). Second, this version is suboptimal as every process involved in the reconstruction must have access to the entire aligned projection series. This limits the number of CPUs we can effectively utilize as the time it takes to transfers large data will largely negate gains in computation. To avoid large transfer of data we plan to add to the Pegasus-managed workflow a pre-processing step that divides the aligned projection series horizontally along the Y axis and have each process reconstruct a horizontal strip of a subset of contiguous Z slices. Also we have also begun to test the employment of predictive submissions, with Pegasus, where a heavier load is placed on more capable processors. Third, as previously mentioned, conversion of the MATLAB code to C/C++ binaries will further increase the speed of the computation.

V. Discussion

We have been pleasantly surprised at the efficiency of moving from MATLAB-based research and development codes to a parallel Grid-based solution. Historically, these advanced mathematical algorithms have been developed using MATLAB in a serial fashion. MATLAB is utilized for its obvious advantages in developing mathematical algorithms. Programs were written serially because the charge of the mathematician is not to develop parallel, Grid-based applications, but rather novel algorithms. Unlike previous Grid-application undertakings, it took relatively little time to create a Grid-based version of TxBR when compared to the development of the actual algorithm. Previous efforts have required almost equal time spent on the development of the algorithm and the Grid-enablement. Moreover, in this case the parallel code was developed by a domain scientist with little knowledge of the Grid.

It used to take several months to convert a MATLAB based set of programs to a parallel Grid-based application. By utilizing a modular (and resource transparent) framework, it can now accomplished (using real world data) in a matter of just a couple of weeks (or even days). Streamlining the

development process in such a manner helps take some load off of our mathematics and development staff so that they can focus on perfecting the cutting edge math they're implementing. By first encapsulating our MATLAB level codes into a Grid solution, we're able to significantly accelerate our "time-to-Grid". This "time-to-Grid" acceleration not only benefits our end-users, but also serves as a platform for the mathematicians to be able to debug and test their programs on real data without have to wait days/weeks for results from serial MATLAB programs.

We are currently developing a workflow to move away from the un-optimized, naively parallel version now utilized. We are also constructing workflows for the 2nd stage of the reconstruction process. We believe that with a new parallelized routine (in a compiled language), coupled with more CPUs, we can easily compute a standard volume in a few hours and smaller volume within minutes, with throughput increase in a manner that is roughly linear to the number of processors tasked.

One of the driving forces for the Telescience Project is to develop a real-time feedback scenario for instrumentation. In this scenario, the Grid is not only utilized to compute data faster but also to refine the collection of raw data. Electron tomography (like other data collection processes) involves a great deal of time-consuming trial-and-error. Due to the nature of the data collection process, researchers are never 100% sure that the data collected from the instrument is actually from the desired region-of-interest (ROI) until after the processing and the visualization of the 3D model generated of that area. This leads to wasted time, and loss of overall sample quality as the electron beam degrades the specimen. If we could calculate smaller volumes of the ROI (during live instrument use) and give the researcher a 3D volume representation of the ROI, we could greatly limit the overall collection time and ultimately increase the fidelity of the raw data.

The developments described here clearly demonstrate that access to computational resources for applications is no longer a bottleneck in the establishment of a real-time Grid. Network performance, however, still remains a critical hurdle. TxBR generates approximately three to four times the amount of output data compared to input data. This means, for example, that a relatively small amount of raw data (10GB) can generate a relatively significant amount of output data (40GB). Distributed across 128 processors, the computation time alone would be approximately 2 hours, not including any queue wait times (a second bottleneck which can be significant). Considering an average network performance of approximately 100Mb/s (with real world disk I/O and TCP network sharing overhead) to external NCMIR computational resources (such as TeraGrid), it will take nearly 1.5 hours to transfer the input data to the computational end point and 3 hours to return the output to the data origin. In this example, over 50% of the end-to-end process time is due solely to data transfers. The problem, however, is that this data transfer overhead is conserved as the level of parallelism is increased. In other words, even if TeraGrid could compute the entire job instantaneously the data transfer overhead (approximately 4.5 hrs in this example) would be preserved. The time necessary to transfer this amount of data is unwieldy and partially or totally negates any gains in computational times attained by using TeraGrid resources.

Another consideration that is worth mentioning, but perhaps outside the scope of this paper, is the issue of distributed software licensing. Here we described the use of MATLAB, which is issued as part of a campus wide license at UCSD. Had this not been the case, the logistics of rights to their licenses between VOs (i.e. TeraGrid) would have to be examined (or at least negotiated). This issue remains a challenge facing the Grid community.

VI. Conclusion

Workflow tools were originally designed to bring together multiple existing applications instead of building a single monolithic application. While that remains true, we have also found that classes of workflow tools are useful and capable of managing the intra-application transaction requirements of parallel applications. Using that capability we have been able to develop MATLAB based parallel codes with relative ease. This is a significant development as the Grid becomes available to applications developers that are outside of the established Grid community.

VIII. References

[1]. Hadida-Hassan, M., Young, S. J., Peltier, S. T., Wong, M., Lamont, S. P., and Ellisman, M. H. (1999) Web-based Telemicroscopy. J. Struct. Biol. 125:2/3, April/May, pp. 235-245

[2]. Molina, T., Yang, G., Lin, A.W., Peltier, S. and Ellisman, M.H. (2005) A Generalized Service-Oriented Architecture for Remote Control of Scientific Imaging Instruments, Proceedings of The 1st IEEE International Conference on e-Science and Grid Computing, p. 56-63

[3]. Peltier ST, Lin AW, Lee D, Mock S, Lamont S, Molina T, Wong M, Martome ME, Ellisman MH (2003) The Telescience Portal for Advanced Tomography Applications. Journal of Parallel and Distributed Applications, Special Edition on Computational Grids, 63(5): 539 - 550

[4]. Lin, A.W., Dai, L., Mock, J., Peltier, S. and Ellisman, M.H. (2005) The Telescience Tools: Version 2.0, Proceedings of The 1st IEEE International Conference on e-Science and Grid Computing, p. 56-63

[5]. Lee, D., Lin, A.W., Hutton, T., Akiyama, T., Shinji, S., Lin, F.P., Peltier, S. and Ellisman, M.H. (2003) Global Telescience Featuring IPv6 at iGrid2002. Future Generation of Computer Systems, 19(6): 1031 - 1039

[6]. BIRN: Biomedical Informatics Research Network, 2005, http://www.nbirn.net

[7]. GEON: Geoscience Network, 2005, http://www.geongrid.org

[8]. Lin, A.W., Dai, L., Ung, K., Peltier, S. and Ellisman, M.H. (2005) The Telescience Project: Applications to Middleware Interaction Components, Proceedings of The 18th IEEE International Symposium on Computer-Based Medical Systems, p. 543 – 548

[9]. GridSphere Portal Framework, 2005, http://www.gridsphere.org

[10]. Portlet Specification, 2003, The Java Community Process, JSR 168, http://www.jcp.org/aboutJava/communityprocess/review/jsr168/

[11]. Jetspeed, 2005, http://portals.apache.org/jetspeed-1/

[12]. Condor Project, 2005, http://www.cs.wisc.edu/condor/

[13]. [Deelman et al 03a] Ewa Deelman, Jim Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, Kent Blackburn, Albert Lazzarini, Adam Arbree, Richard Cavanaugh, and Scott Koranda. "Mapping Abstract Workflows onto Grid Environments", Journal of Grid Computing, Vol. 1, No. 1, 2003.

[14]. [Deelman et al 03b] Ewa Deelman, Jim Blythe, Yolanda Gil, and Carl Kesselman. "Workflow Management in GruPhyN", In Grid Resource Management, J. Nabryski, J. Schopf, and J. Weglarz (Eds), Kluwer 2003.

[15]. [Deelman et al 03c] 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, Daniel S. Katz, Pegasus: a Framework for Mapping Complex Scientific

Workflows onto Distributed Systems, Scientific Programming Journal, Volume 13, Number 3, 2005 [Deelman et al 04] "Pegasus: Mapping Scientific Workflows onto the Grid", Ewa Deelman, Jim Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, and Miron Livny. Across Grids Conference, Nicosia, Cyprus, 2004

[16]. Lin, A.W., et al., The Telescience Project: Transparent Grid Access for Scientific Communities, Special Issue of Concurrency and Computation: Practice and Experience – Science Gateways at GGF14 (Submitted)

[17]. Lawrence, A., Bouwer, JC., Perkins, G., and Ellisman M.H. (2005) Transform Based Backprojection for Volume Reconstruction of Large FormatElectron Microscope Tilt Series, Journal of Structural Biology (Accepted forPublication December 2005)

[18]. Lawrence A., Bouwer, J., Perkins, G., Kulungowski, A., Peltier, S., and Ellisman, M.H. (2005) Electron Microscope Tomography: Calculating and Inverting theGeneralized Ray Transform, Proceeding of the SIAM Conference onImaging Sciences, May 15-17, 2006, (Submission accepted December 2005)