Building Solutions for Researchers with a Science Gateway

John McGee, Jason Reilly, Mats Rynge
Renaissance Computing Institute (RENCI)
University of North Carolina at Chapel Hill

1. Introduction

One important goal for the TeraGrid (TG) Science Gateways program[1] is to bring the capabilities of this national cyberinfrastructure (CI) to the forefront of scientific discovery and scholarly efforts of a broad and growing community of researchers, students, and educators. The TG Gateway program is addressing a critical need for greater accessibility of the national resources for researchers from many science domains, with a wide distribution of information technology (IT) skills, experience, and most importantly, level of interest. The RENCI Engagement Team has enjoyed a broad perspective of these challenges through our participation in the TeraGrid Science Gateway effort, and by leading the Engagement program for the Open Science Grid (OSG)[2].

First generation TeraGrid science gateways were primarily web portal applications integrated with TG resources for computation, data analysis, and visualization. In our experience, the usage model of a web portal has proven to be valuable in education and exploratory scenarios, but has also proven to be very unlikely to achieve a sustained and meaningful impact on researchers working towards scientific discovery. These researchers typically have an existing work process pipeline for their applications and data running on lab, departmental, and/or campus level resources, which they are reluctant to recreate or recast around a web user interface.

Guided by these efforts, we have developed a technology platform, and a methodology of working closely with researchers over a short period of time to bridge the gap between user readiness and CI capabilities on an individual basis. We have determined that broad adoption of CI concepts and technologies most often requires an *embedded* effort to develop the skills of students, faculty, and the IT organizations of the campuses that host them. Both infrastructure providers and consumers must be *immersed* in building and delivering the end-to-end processing and data management capabilities needed for scientific discovery. We have recently begun applying this Embedded Immersive Engagement for Cyberinfrastructure (EIE-4CI)[3] methodology to the RENCI TeraGrid Science Gateway, which has changed our perspective and development path for the Gateway.

By implementing EIE-4CI with the TeraGrid Gateway, we consider the portal, infrastructure, and services as a toolbox to be used in the engagement process, not as a turnkey solution. We work together with researchers to understand their usage models, embed gateway services into their work process pipeline, and assemble new capabilities as needed. Each engagement enriches the toolbox for the next, and builds important relationships between CI providers and consumers, facilitating mutual trust and understanding. These relationships are critical to evolving the fundamental CI requirements and ensuring that the national cyberinfrastructure maintains a sustainable and growing impact in the research and educational technology ecosystem. In this paper, we describe the architecture of the RENCI Science Portal, a technology platform that the Engagement Team uses to rapidly assemble powerful solutions for researchers backed by the nation's most capable cyberinfrastructures. We also describe two example engagements using this platform, and address some open issues and future plans.

2. The RENCI Science Portal Architecture

2.1 Overview

The RENCI Science Portal provides a highly scalable platform to rapidly assemble TeraGrid powered solutions for researchers. Figure 1 below diagrams the architecture of the platform. One way to think of a science gateway, is as a broker of value added services between CI providers and CI consumers. As such, the RENCI Science Portal platform (the platform) interfaces with CI providers using the necessary advanced Grid technologies and protocols, yet the flexible and numerous interfaces for CI consumers are commodity and traditional IT based technologies and protocols, enabling broader adoption, and integration with a wide array of open source and commercial tooling that may be in use by the CI consumers.

2.2 Generating Platform Components From Application Metadata

The North Carolina Bioportal[4], the predecessor to the RENCI Science Portal, was based on the PISE[5] XML tool, a collection of metadata captured in XML detailing BioInformatic command line applications. It was used to build and run CGI scripts which were submittable over http requests. PISE is generative, meaning that to add new command line applications, all that is required is to write a new XML file based on a schema definition, then run the built-in generators. There were however some limitations in the implementation, including I/O overhead for reading and validating the xml at runtime, shelling out to run perl scripts (parameter evaluations) at runtime, difficulty in maintaining the XML schema, generative tasks are build-time only, and discontinuity of programming languages. Finally, PISE development seems to have been discontinued as of 2003. The new platform builds extensively upon this generative philosophy to rapidly introduce new applications and capabilities, and enable such a large and complex system to be managed and supported with surprisingly little effort.

The Annotations[6] feature was introduced to the Java programming language starting with version 1.5, and is an interface that enables developers to add metadata to software code. Java Annotated code can be used for compile-time, deployment-time or runtime information processing. The new platform has replaced PISE XML with a set of customized Java Annotations. Replacing PISE allows for a more flexible source of command line application metadata, and maintenance is improved due to the strongly typed nature of Java, and compile-time errors are exposed immediately. Integrating additional command line applications is less rigid since altering the annotations themselves is part of the same code base.

The RENCI Science Portal infrastructure builds upon these annotations to generate various secure accessors[7] to launch command line applications. This includes generating JSR-168 compliant portlets, asynchronous and synchronous web services, and BioMoby[8] web services. These annotations are also used at runtime by a number of Java Swing desktop applications, and to generate documentation for the web portal, TG Gateway Web Service Registry[9] metadata, and application requirements that are used in the MatchMaking[10] process. There are currently 127 science applications available by these various mechanisms within the RENCI Science Gateway. Based on demand, adding new generative modules would enable the introduction of additional access mechanisms across all of these applications (eg Google Gadgets or other Web 2.0 client technologies).

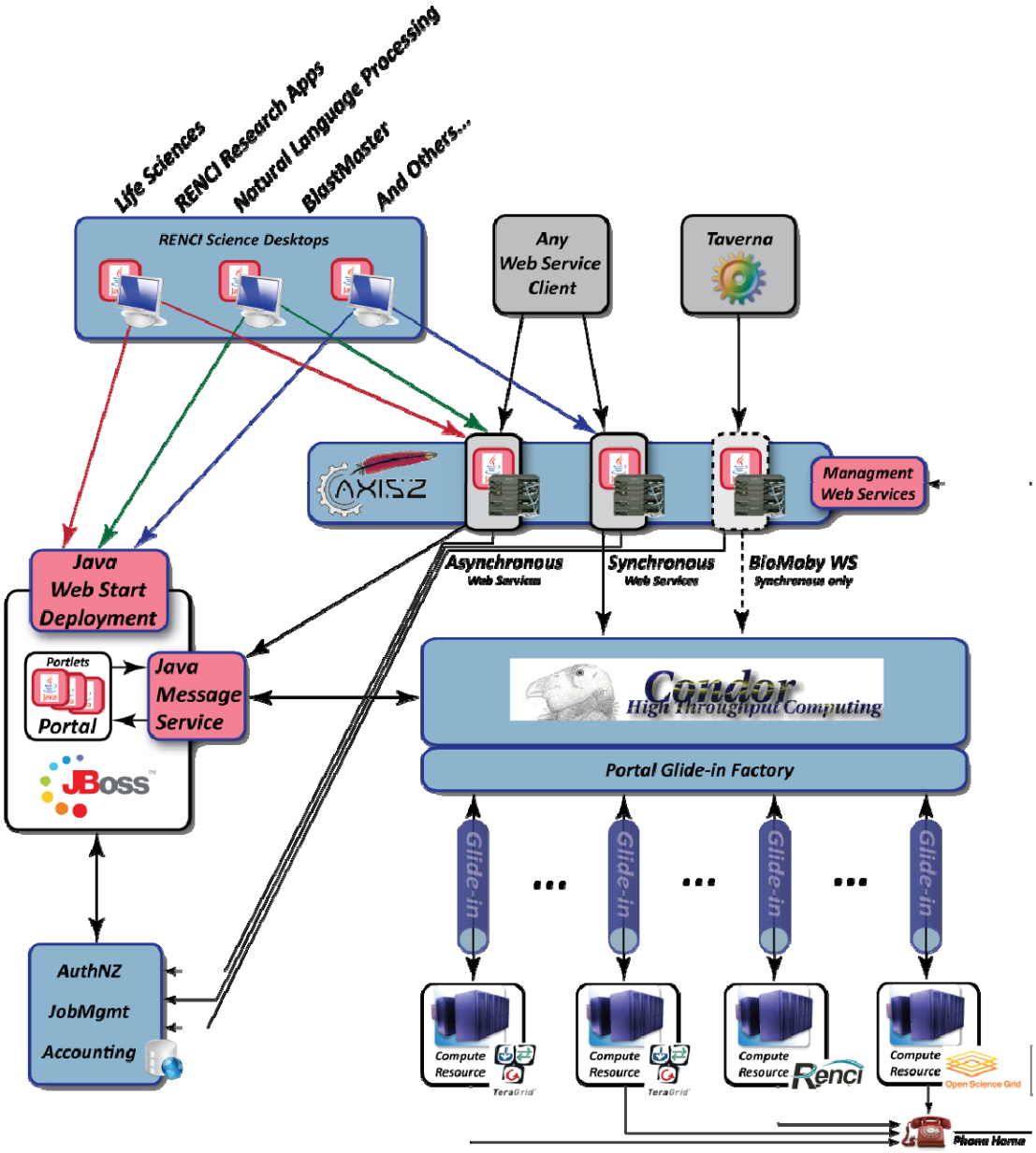

Figure 1: RENCI Science Portal Platform Architecture

2.3 Hosted Service Layer

The web service layer acts as an integral part of the RENCI Science Portal infrastructure. The generated web services authenticate against the platform account management system and are available to be called securely over an HTTPS transport in one of two ways: asynchronously or synchronously. An asynchronous web service invocation returns a jobId, and upon invocation, persists to a database any parameters and files associated with that job. The new record in the database generates a unique jobId, which is sent as a Java Message Service (JMS) ObjectMessage to a Topic deployed on a JBoss Portal server. Before acknowledging the new message in a configured JMS MessageListener, the job is wrapped in a Runnable thread and executed via a ThreadPoolExecutor. This Runnable is comprised of three tasks. The first task is to submit the job to a local Condor[11] queue using BirdBath[12]. The second task is to monitor the state of that job and persist the status back to the database when the

job changes to a done or failed status. And lastly, the Runnable persists output information. For these asynchronous invocations, a user can make separate synchronous web service calls to check the status of a submitted job as well as download the results programmatically when the job has completed.

Invoking synchronous web services in the RENCI Science Portal behaves much in the same way as the asynchronous web services except that instead of sending a JMS message to a Topic, the service submits directly to the local Condor queue. With the synchronous invocation, the client connection to the web service must be maintained for the web service to successfully return the output. If the client connection is broken, the service will continue to run and the user can download the results by logging into the Portal with a web browser, however the client that invoked the synchronous service will not be able to access the results.

2.4 Science Desktops

The RENCI Science Desktops are a set of Java Swing client applications, launched via Java Web Start (JWS) through a web browser. The Science Desktops provide a rich user interface enabling researchers to securely interact with the Gateway from their desktop. All interactions with the gateway layer require the user to present their RENCI Science Portal username and password each time the JWS desktop application is launched (credentials are not persisted on the client), and all data transfers occur over a standard secure socket layer (SSL, port 443). The current set of Science Desktops include: BLASTMasterDesktop, NLPDesktop, RENCIDesktop, and LifeSciencesDesktop, and they operate by calling into the asynchronous web services of the platform to submit jobs, and the synchronous web services to monitor job status and download results when the jobs have completed. These specialized interfaces are relatively easy to create due to: the reliance upon the gateway service layer; application specific aspects are generated from the Java Annotations as described in section 2.2; and code reuse - general behaviors such as authentication, client side preferences, multithreaded file upload, and multi-job submissions, are shared among the various desktops. These desktop applications enable us to provide a highly specialized environment that delivers a very specific thin slice of the national cyberinfrastructure capability to researchers, as motivated by the use case in section 3.1 below.

2.5 Submitting Jobs to TeraGrid, Open Science Grid, RENCI, UNC-CH Resources, and Beyond

All jobs coming into the platform are loaded into the Condor pool, which starts off as a pool with zero resources, and resizes dynamically (in both directions) using Condor glide-ins[13] based on the pending workload. The glide-in factory is the component of the platform that manages the glide-ins and is loosely based on the GlideinWMS[14] system. It consists of a daemon monitoring the state of the gateway job queue, the glide-in job queue, and the state of submitted glide-ins within the system. When the glide-in factory determines that more glide-ins are needed, available resource descriptions are read in from a database, and the resource set is narrowed down to a subset based on the requirements of the job. For example, if a job states that it requires science applications and file databases, only resources which can meet those requirements are kept in the resource set. Note that the assertion that an application has such a requirement originates with the application metadata via Java Annotations, and that any required applications and databases are pre-installed by the Engagement Team on the target resources. Next, each resource is assigned a score based on the state of current glide-ins at the resource, a multiplication factor used to prefer resources over others, and an assigned max allowed glide-ins for the resource. The resource with the highest score will be the one that the glide-in is submitted to as a Condor-G job, and ultimately as a GRAM2[15] job.

Glide-ins can be sent to the TeraGrid, Open Science Grid, BASS[16] (an NIH sponsored resource in the computer science department at UNC Chapel Hill), and local resources at RENCI. Each infrastructure has its own set of policies that must be accounted for as decisions about new glide-ins are made. One policy example is that BASS will only service users who have a grant funded by NIH, so the gateway has a mechanism to capture voluntarily offered NIH award numbers and will include/exclude BASS from scheduling decisions based on the availability of an NIH award number from the user that is interacting with the gateway (award numbers are maintained in the Gateway user profile database). There are additional factors that the glide-in factory considers, including expected availability, expected runtime, and how many jobs are currently in the system. Jobs that are known to have a short runtime are easily directed to local RENCI resources avoiding long queue wait times on the in-demand national resources, and a submission of a very large number of jobs can be spread across TeraGrid, OSG, BASS and RENCI to achieve the highest possible throughput, assuming access and policy compliance. Additional backend engines can be added, including commercial providers such as Amazon EC2 and Microsoft Azure, which would be completely transparent to the end users.

2.6 Late Binding of Jobs to Resources

Keeping the gateway and the job execution layers as two separate entities has enabled late binding of jobs to resources. Neither the user nor the gateway layer should have to know (or care) where the job will run. The burden of fault tolerance and issues such as a long queue wait time is the responsibility of the execution layer (Condor and the glide-in factory) and is hidden from the user and gateway layer. Late binding, in different forms, has proven to solve many usability problems and is becoming popular on both TeraGrid and OSG via systems such as: MyCluster[17], Falkon[18], Pegasus Corral[19], GlideinWMS, and PanDA[20].

Fault tolerance is an important requirement for a Science Gateway, and is especially important to handle as automatically as possible to mitigate troubleshooting efforts by gateway staff. Glide-ins and late binding provide a first pass on verifying that a resource is currently fit for jobs. If the glide-in is stuck in a queue, or fails to set up the environment and start, no jobs will be bound to the resource, and from the point of view of the user and gateway, it is like the resource does not even exist. As with any complex, distributed, heterogeneous environment, systems will be unavailable due to both scheduled and unscheduled events including: power outages, file system failures, software problems, system maintenance, limited user disk space, network connectivity problems, etc. A cursory review of the news@teragrid.org list for the month of April, 2009 indicates that: Abe was paused 7 times & had scheduled maintenance once; Kraken was paused once, had unscheduled maintenance once, and rebooted 6 times due to FS (Lustre) issues; Ranger had FS issues once; Queen Bee had FS issues once; ORNL NSTG was down once for system maintenance; BigRed was down once due to system maintenance; Pople was down once due to system maintenance; Cobalt was down once for scheduled system maintenance and once for unscheduled maintenance; Mercury was down once for unscheduled maintenance; and all SDSC clusters were inaccessible for four hours due to router maintenance. Most of these issues can be detected and handled automatically within the glide-in factory, without any science gateway staff intervention. The science gateway users does not even know that issues like these occur behind the gateway.

Because of this late binding, the per-job gateway auditing information requirements for the community account cannot be captured until after that job has landed at a target resource. We also wish to capture metrics information using kickstart[21], which similarly must be collected and aggregated back to the gateway management database as part of the remote job execution. The platform accomplishes this by wrapping the compute job with a

"phone-home" client as depicted in Figure 1 that invokes a gateway management web service as part of the job execution, notifying the gateway of exactly where and when the job ran and how it performed. The phone-home feature enables the platform to record in the job management layer exactly which TeraGrid resource was used for every job, and performance metrics accumulated along the way are used by the glide-in factory to influence future resource selection decisions.

2.7 Grid Credential Management

Another task for the glide-in factory is to manage X.509 credentials. Each of the backend infrastructures are using X.509 grid credentials for authentication, but have different requirements on the extra information that must be included in the proxies. For TeraGrid, the glide-in factory loads a TeraGrid Community Credential and adds a set of SAML attributes[22] including portal user name, portal user email address, and authentication timestamp. For OSG, voms-proxy-init is used to generate a proxy, again with SAML attributes, but in this case the attributes are derived from the VO Management Service (VOMS)[23]. The platform currently does not submit to OSG sites that require gLExec[24], as the Science Portal users do not have individual X.509 user certificates, which are required for sites that require glExec. For BASS and local RENCI systems, plain X.509 proxies are used without SAML attributes.

3. Example Engagements

3.1 Metagenomics: Generating Too Much Data to BLAST Locally

The RENCI Engagement team has worked with a metagenomics collaborative[25] seeking to identify new viruses. This team contacted the RENCI Science Gateway team to learn about gaining access to TeraGrid resources to help with a computational quandary. With the accessibility of genome sequencers such as those available from 454 Sequencing[26], research labs are increasingly generating data at a rate beyond their capacity to process it locally. In discussing their needs, together we determined that the best approach would be for us to develop a new Java Web Start client application. This new science desktop would enable them to: 1) to point at a directory containing any number of zip files, each containing hundreds of files, each file containing a sequence, each sequence representing a single BLAST job; 2) configure the BLAST execution parameters; and 3) simply "say go". This collaborative regularly needs to process 100k+ BLAST jobs, has NSF/NIH funding, and a successful track record of identifying new viruses and new scientific techniques. The science desktop developed during this engagement has been labeled the BLASTMaster[27]. Figure 2 below shows how the jobs were distributed by the execution layer of the platform for one particular study conducted by the collaborative using BLASTMaster. In this case, the 96,866 individual sequence files were grouped into roughly 480 zip files, which were transferred to the Gateway for execution by the science desktop.

The determination of the approach to this engagement is a key point of the methodology, as we had walked through a number of other potential usage models together with the researchers, with varying levels of integration-with and interruption-of their existing process pipeline utilizing local resources. The negotiation process is critical to ensure the best possible chance for a sustained impact, and to identify opportunities for high leverage and impact. In talking with a number of other researchers, we believe that the BLASTMaster has a high probability of evolving into widely used tool, and being responsible for meaningful impact.

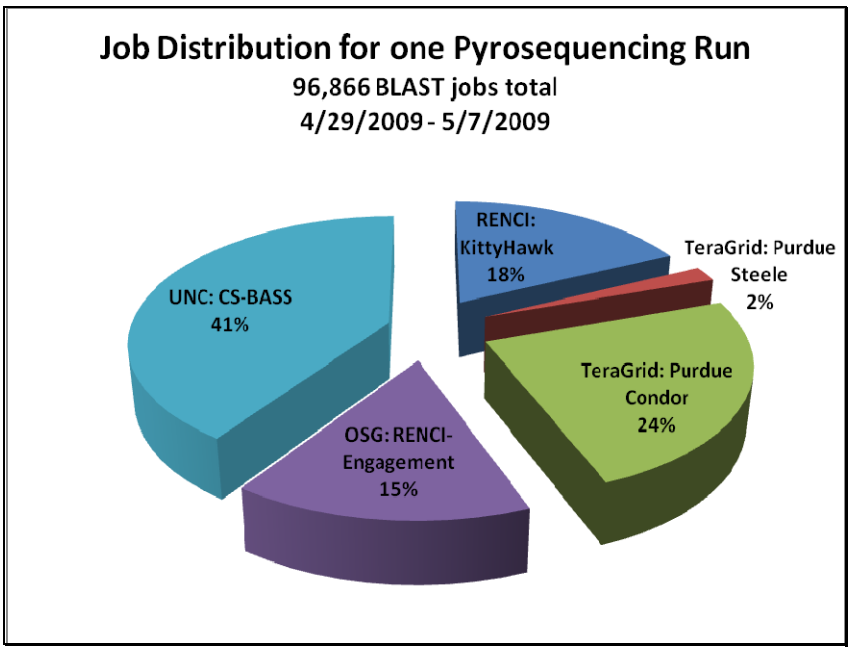

Figure 2: Distribution of BLAST jobs from one Pyrosequencing run, with the glide-in factory configured for: three TeraGrid resources, one OSG resource, one RENCI resource, and one UNC-CH resource

The BLASTMaster Java Web Start desktop application relies upon the existing RENCI Science Portal infrastructure including the asynchronous web services, user management, TeraGrid integration, etc. New capabilities that required development for this engagement included multi-threaded job submission, monitoring, and results fetching. These capabilities were built upon existing services in the platform to manage a large number of jobs with a single submission. These new science desktop features are wholly applicable to any other application within the gateway that shares this usage model, and is an example of a needs driven enhancement to the platform.

This one user engagement has the potential to quickly burn through the TeraGrid allocation that has been awarded to the RENCI Science Portal in the form of a community account. However, having the community account allocation was absolutely critical to lowering the barrier to entry for this collaborative, and enabled us to quickly demonstrate value. As production runs ramp up, we must monitor allocation usage, and develop a plan for helping this team become self sustaining. This could include: helping them through the process of acquiring their own allocation which could be added into the Gateway for their use only; some form of new and unique linkage of allocations and access that allows a gateway to operate on behalf of a user; a new mechanism allowign a user to delegate a portion of their allocation to a gateway community account; or some new form of allocation made to a gateway on behalf of a specific end user. There are a number of non-trivial issues yet to be resolved in providing gateway services to users with their own credentials, such as the differences in the environment that is established for the jobs under the users credential versus the gateway credential.

3.2 Natural Language Processing

The RENCI Engagement Team was approached by a researcher using Natural Language Processing techniques over a corpus of 160k documents from the medical research domain. The challenge was that the computations could not be completed in a timely manner for the research and education requirements. In this case, there were no new additional features required of the platform except to add the Stanford Parser[28] as a supported application, and configure the deployment scripts that maintain applications in the RENCI community software area across the TeraGrid resources. The end result was that in just a couple of days, the widely used Stanford Parser is now available on the TeraGrid for large scale studies via: web services, portlets, and an easy to use Java Web Start client application. Additionally, this application is now registered into the emerging TeraGrid Gateway WS Registry, thus broadcasting its availability to a wider audience as the TG WS Registry becomes searchable and more widely used.

4. Conclusion and Future Plans

The leading goals for the RENCI Engagement Platform described in this paper include: scalability, maintainability, policy aware workload/resource matching across disparate infrastructures, and most of all, accessibility and ease of use for the research communities[29]. Coupled with our expertise in negotiating technical solutions with researchers, we believe that this technology base provides a unique capability for rapidly ramping up computationally based research from local lab/campus levels, to a national scale. Because the EIE-4CI methodology relies upon working closely with individual researchers and small teams, we are careful to guide any non-trivial platform enhancement efforts to projects where there is evidence of strong leverage and broad applicability.

Future plans include investigating the meta-scheduling services emerging for the TeraGrid, such as the Condor-G MatchMaking support[30]. We will also address an issue with the gateway auditing and accounting requirements coupled with the currently available accounting mechanisms. The RENCI Gateway sends the required gateway auditing information into the TeraGrid with the job itself in the form of SAML assertions as described in section 2.5 above. This is a very convenient mechanism in that the gateway has fulfilled all of its per-job auditing responsibilities with the job submission itself. However, this becomes difficult when using glide-ins in combination with a TeraGrid Community Account. To maintain compliance with community account policies, we have implemented a policy whereby a glide-in to a TeraGrid resource can be used by exactly one gateway user, and that gateway user's information is embedded into the glide-in startup job via the SAML assertions. This limits many benefits of using glide-ins, and the negative effect of this limitation increases as the number of simultaneously active unique gateway users increases. We plan to pursue a dialog within the TeraGrid Gateway community to assist in determining a reasonable solution.

Acknowledgements

We would like to thank Susan Miller, Chunlin Wang, and Eric Delwart for their direction and patience in refining the BLASTMaster desktop application. We would like to thank Cathy Blake for her work in developing the NLPDesktop (Stanford Parser). Finally, we would like to thank Edward Dale and Russell Taylor for providing access to the UNC Computer Science BASS resource, and deploying new services to support our platform framework. This research was done using resources provided by the TeraGrid, Open Science Grid, and BASS which are supported by the National Science Foundation, the U.S. Department of Energy's Office of Science and the National Institutes of Health.

References:

[1] N. Wilkins-Diehr, D. Gannon, G. Klimeck, S. Oster, S. Pamidighantam, "TeraGrid Science Gateways and Their Impact on Science", IEEE Computer, Volume 41, Number 11 (November, 2008), pages 32-41.

[2] https://twiki.grid.iu.edu/twiki/bin/view/Engagement/WebHome

[3] The Embedded Immersive Engagement for Cyberinfrastructure (EIE-4CI) methodology is derived from RENCI's experience in the Open Science Grid (OSG) Engagement program, and is currently funded by NSF CI-TEAM

[4] Ramakrishnan, L., M. S.C Reed, J. L. Tilson, D. A. Reed "Grid Portals for Bioinformatics" Second International Workshop on Grid Computing Environments (GCE), Workshop at SC|06, Tampa, Florida (2006). http://www.renci.org/wp-content/pub/ncbioportal/BioportalPaper-GCEFinal.pdf

[5] http://www.pasteur.fr/recherche/unites/sis/Pise/
[6] http://java.sun.com/docs/books/tutorial/java/javaOO/annotations.html
[7] http://en.wikipedia.org/wiki/Method_(computer_science)
[8] http://biomoby.open-bio.org/
[9] http://www.teragridforum.org/mediawiki/index.php?title=Science_Gateways_WSInfo
[10] http://www.cs.wisc.edu/condor/manual/v7.2/2_3Matchmaking_with.html
[11] http://www.cs.wisc.edu/condor/
[12] http://www.cs.wisc.edu/condor/birdbath/
[13] http://www.cs.wisc.edu/condor/manual/v7.3/5_4Glidein.html
[14] http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/
[15] http://www.globus.org/toolkit/docs/4.0/execution/prewsgram/
[16] http://wwwx.cs.unc.edu/Research/bass/index.php/Main_Page
[17] http://www.tacc.utexas.edu/mycluster/
[18] http://dev.globus.org/wiki/Incubator/Falkon
[19] http://pegasus.isi.edu/corral/latest/
[20] https://twiki.cern.ch/twiki/bin/view/Atlas/Panda
[21] http://pegasus.isi.edu/publications/kickstart.pdf
[22] http://www.teragridforum.org/mediawiki/index.php?title=Science_Gateway_Credential_with_Attributes
[23] http://edg-wp2.web.cern.ch/edg-wp2/security/voms/voms.html
[24] http://vdt.cs.wisc.edu/components/glexec.html
[25] Eric Delwart: http://cancer.ucsf.edu/people/delwart_eric.php
Chunlin Wang: http://med.stanford.edu/sgtc/general/staff.html#wz
[26] http://www.454.com/
[27] https://portal.renci.org/BLASTMasterDesktop/
[28] http://nlp.stanford.edu/software/lex-parser.shtml

[29] Michael Litzkow and Miron Livny, Experience with the Condor Distributed Batch System; Proceedings of the IEEE Workshop on Experimental Distributed Systems; (October, 1990), Huntsville, AL
http://www.cs.wisc.edu/condor/doc/experience.pdf

[30] http://www.teragridforum.org/mediawiki/index.php?title=Schedwg_condorg_userguide