Introducing The Neuroscience Gateway

Subhashini Sivagnanam, Amit Majumdar,
Kenneth Yoshimoto, Vadim Astakhov,
Anita Bandrowski, MaryAnn Martone
University of California San Diego
La Jolla, CA, USA

Nicholas. T. Carnevale
Yale School of Medicine
New Haven, CT, USA

***Abstract*— Last few decades have seen the emergence of computational neuroscience as a mature field where researchers are interested in modeling complex and large neuronal systems and require access to high performance computing machines and associated cyberinfrastructure to manage computational workflow and data. The neuronal simulation tools, used in this research field, are also implemented for parallel computers and suitable for high performance computing machines. But using these tools on complex high performance computing machines remain a challenge due to issues with acquiring computer time on these machines located at national supercomputer centers, dealing with complex user interface of these machines, dealing with data management and retrieval. The Neuroscience Gateway is being developed to alleviate all of these barriers to entry for computational neuroscientist. It hides or eliminates, from the point of view of the users, all the administrative and technical barriers and makes parallel neuronal simulation tools easily available and accessible on complex high performance computing machines and handles the running of jobs and data management and retrieval. This paper describes the architecture it is based on, how it is implemented, and how users can use this for computational neuroscience research using high performance computing at the back end.**

Keywords—computational neuroscience, science gateway, high performance computing

I. INTRODUCTION

Computational neuroscience has seen tremendous growth in recent years. This is evident from the large number of publications, in prestigious neuroscience journals, that are more and more based on modeling. In the last two decades, this has motivated development of simulation tools such as NEURON [1], GENESIS3 [2], MOOSE [3], NEST [4], PyNN [5] and Brain [6] which are mature and written for parallel computers. Complex neuroscience problems, which involve network models, optimization or exploration of high dimensional parameter spaces, require access to high performance computing (HPC), data storage and complex workflow. Yet accessing and using HPC resources remain difficult due to the need to write yearly allocation proposals for acquiring computer time on national supercomputer centers, understand complex HPC architecture, learn complex OS/software, optimally install neuronal software, learn policies and batch environments, manage data transfer/retrieval, and understand remote authentication issues. As a solution to this problem we have developed a community infrastructure layer, i.e. a science gateway [7], specifically for computational neuroscientists, that abstracts away technical and administrative details of underlying hardware. As a result this allows neuroscience researchers easy access to best neuroscience simulation and analysis packages running on large scale HPC resources. The neuroscience gateway (NSG) [8] started its friendly user phase since December 2012. As a part of this we introduced some users, to NSG, who were recruited earlier to help us test the NSG. Since late December, 2012 we consider it to be in production and we continue to add more neural simulation tools. This paper describes the NSG software architecture, how it is implemented, the impact it has had until now, and the future plan.

II. MOTIVATION AND BACKGROUND

Most computational neuronal modeling projects start "small" and many stay "small," in the sense of being accommodated by individual desktop systems, but many eventually outstrip the speed and/or storage capabilities of local hardware. This is most often the case with projects that involve complex models (especially large scale networks) or complex protocols (often involving learning rules), or require high-dimensional optimization or parameter space exploration. Such projects have a tremendous potential to use cyberinfrastructure (CI), but only a very few neuroscientists have been able to perform simulations on extreme scale HPC machines [9, 10, 11]. There is a broad consensus that the wider computational neuroscience community needs access to complex CI and HPC resources. Those who lack such access are at a significant disadvantage relative to the very few who have it. This disparity is particularly telling in light of the fact that the widely used simulators such as NEURON, GENESIS3, MOOSE, NEST, Brian and PyNN have been implemented on and released for parallel hardware, including HPC machines, for several years now. The typical neuroscientist - even one who is engaged in computational modeling - has limited compute resources, usually only a few desktop machines. The investigator whose project requires HPC machines must write a yearly proposal for allocation of supercomputer time. In the US these are peer reviewed for computer time on National Science Foundation (NSF)'s HPC machines and there is no guarantee that requested resource will be made available. If the proposal succeeds, the next task is to install simulation software (NEURON, GENESIS 3, MOOSE, NEST, PyNN, Brian) optimally on the supercomputer, then apply the batch software process to configure and run the model on the cluster, and finally deal with output data retrieval and storage issues. These steps involve many vexing details that are not only time consuming

and requires knowledge of HPC and IT, but also differs significantly from facility to facility. Investigators who want to use neuroal software on HPC resources directly, have to deal with these issues themselves. The entire process constitutes a barrier that impedes effective utilization of HPC resources and also distracts neuroscientists from their primary research goals. We believe that many of issues are encountered by neuroscientists in other parts of the world such as in Europe and Asia. Our proposed solution enables the entire community of computational neuroscientists to access NSF (and other) funded national HPC resources transparently through a common, convenient interface that is already configured for optimal use and operates within a common spatial and semantic framework. The benefit of this infrastructure can be extended to many more end users, allowing investigators to focus on their research and fostering collaboration.

In recent years user-friendly scientific gateways have been developed successfully for many research fields and have resulted in tremendous adoption of CI and HPC by the broader user community of those fields. A few examples of such gateways in the US include the nanoHUB gateway [12] for nanotechnology, the CIPRES gateway for phylogenetics research [13], the GRIDCHEM [14] gateway for computational chemistry, and the ROBETTA [15] gateway for protein structure prediction. Similarly many gateways exist in Europe and Asia. In the US moust of these gateways are funded by the NSF for specific scientific domains, and primarily utilize NSF's Extreme Science and Engineering Discovery Environment (XSEDE) [16] or Open Science Grid (OSG) [17] for accessing HPC resources. In addition to NSF funded science gateways, there are gateways that are funded by other organizations such as the US Department of Energy (DOE) at various DOE laboratories. Similarly outside of US, there are many e-infrastructures and gateways available for different domain sciences. Specific to neuroscience there is the neuGRID [18], which offers a science gateway for neuroscientists to facilitate the development of image analysis pipelines using HPC resources aiming at accelerating research in Alzheimer and other neurodegenerative disease markers. The Blue Brain project [19] is further developing its own portal environment.

III. NSG ARCHITECTURE

The NSG architecture design is based on the existing CIPRES Science Gateway framework [20] which has been very successful and popular for building the phylogenetics gateway as well as other gateways such as the PoPLAR (Portal for Petascale Lifescience Applications and Research) [21] gateway. CIPRES is very mature framework, and was implemented at the San Diego Supercomputer Center (SDSC), by SDSC researchers and programmers, using the open source Workbench Framework (WF) [22]. Below we describe the various software architecture components of WF, their functional adaptation specifically for the NSG and the current usage monitoring and metrics of the NSG.

A. Underlying Architecture

The WF is a software development kit (SDK) designed to generically deploy analytical jobs and database searches to a generic set of computational resources and databases. The WF contains modules to manage submission of jobs to analytical tools on various computational resources and modules to manage queries on data resources. The higher level schematic of the WF architecture, as described in a diagram by the WF project [22] developers is shown in Fig. 1. It is the basic software architecture of the NSG. The modules in the WF are as follows:

Presentation Layer: The WF Presentation Layer accesses SDK capabilities through the J2EE front controller pattern, which involves only two Java Classes. As a result, the WF is neutral with respect to interface access. The presentation layer provides lightweight access through a web browser and preserves flexibility for alternative access routes and adopts an architecture based on Linux, Apache Tomcat, MySQL, and Java Struts2. The browser interface is based on the look and feel of popular email clients and supports data and task management in user-created folders. The complexity of gateway layer is hidden from users through this interface that also allows users to create a login-protected personal account. Registered users can store their data and records of their activities for a defined period of time. Uploaded user data is checked for format compatibility. Users can also manually specify data types and formats.

User Module: The User Module passes user-initiated queries and tasks from the interface to the executive portions of the infrastructure via data and task management modules. It also stores all user data and task information in a MySQL database. It supports individual user roles, permitting the assignment of individual user accounts, the sharing of data between accounts, and selective access to tools and data sources that may be proprietary. Mapping user information takes places at this layer which helps track the individual usage on computational resources.

Broker Module: The Broker Module provides access to all application-specific information in a Central Registry. This Registry contains information about all data types required as input and output for each application along with the formats accepted by each tool. Concepts and concept relationships are formulated in XML files and read by Central Registry API implementing classes. Defining tools and data types in a single location allows adding new tools and data types with no impact on the functioning of the application outside the Registry.

Tool Module: The Tool Module manages the translation of tasks submitted by users into command lines and submission of the command line strings along with user data for execution by appropriate compute engines. The Tool module handles data formatting for jobs, and job staging. It also keeps track of which tools can be run on which computational resources, and the status of those resources. The design allows great flexibility in determining what assets the NSG can access for job execution. Computational resources can be added through editing the tool resource configuration file, and the application can send command line scripts and receive output via

essentially any well-defined protocol (e.g. Unix command line, web services, SSH, DRMMA, GRAM, gsissh, etc.).

External Resources: The generic design of the WF architecture supports access to a wide variety of computational resources and databases, whether local or remote. Access can be accomplished through a combination of individual mechanisms, including SSH, GRAM/Globus, SOAP, and ReST services.

B. CIPRES Adaptation for the NSG

The adaptation of CIPRES WF architecture to NSG was done with the idea of hiding all the complexities associated with accessing and using a HPC resource such as job submission, input data transfer, choosing of machine specific HPC parameters, output retrieval and storage etc. Fig. 2 shows the high level functional diagram of this adaptation. Though NSG's initial software design was based on the CIPRES WF architecture, our implementation contained enhancement and modification to the existing software based on the needs of the neuroscience community. Hardware needed for setting up the NSG utilizes SDSC's reference VM server, MySQL, Cloud storage and webservices. The latest software version of WF architecture was obtained from SVN maintained by CIPRES developers. Following is the list of the key adaptations done to the existing CIPRES code base for the NSG.

1. Addition of uuencode/uudecode functionality to support upload of input files in zip format
2. Modification of job submission environment to accommodate compilation of the NEURON code
3. Storing of output file per session in SDSC's Cloud storage
4. Automatic deletion of unused user files based on time length of inactivity
5. Define computational neuronal tools in the PISE XML format and interface with the portal

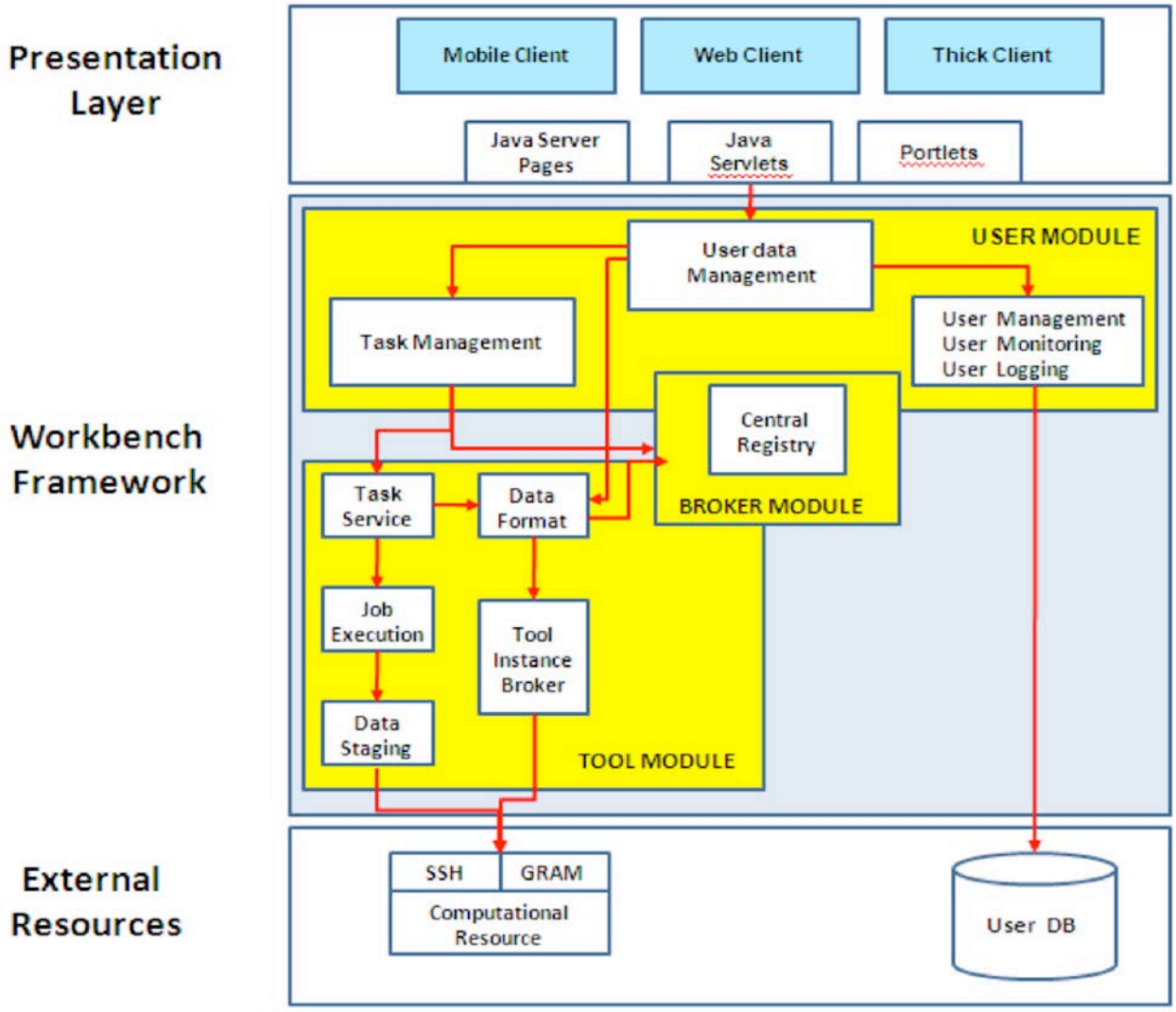


Fig. 1. Workbench Framework (from WF [22] project page).

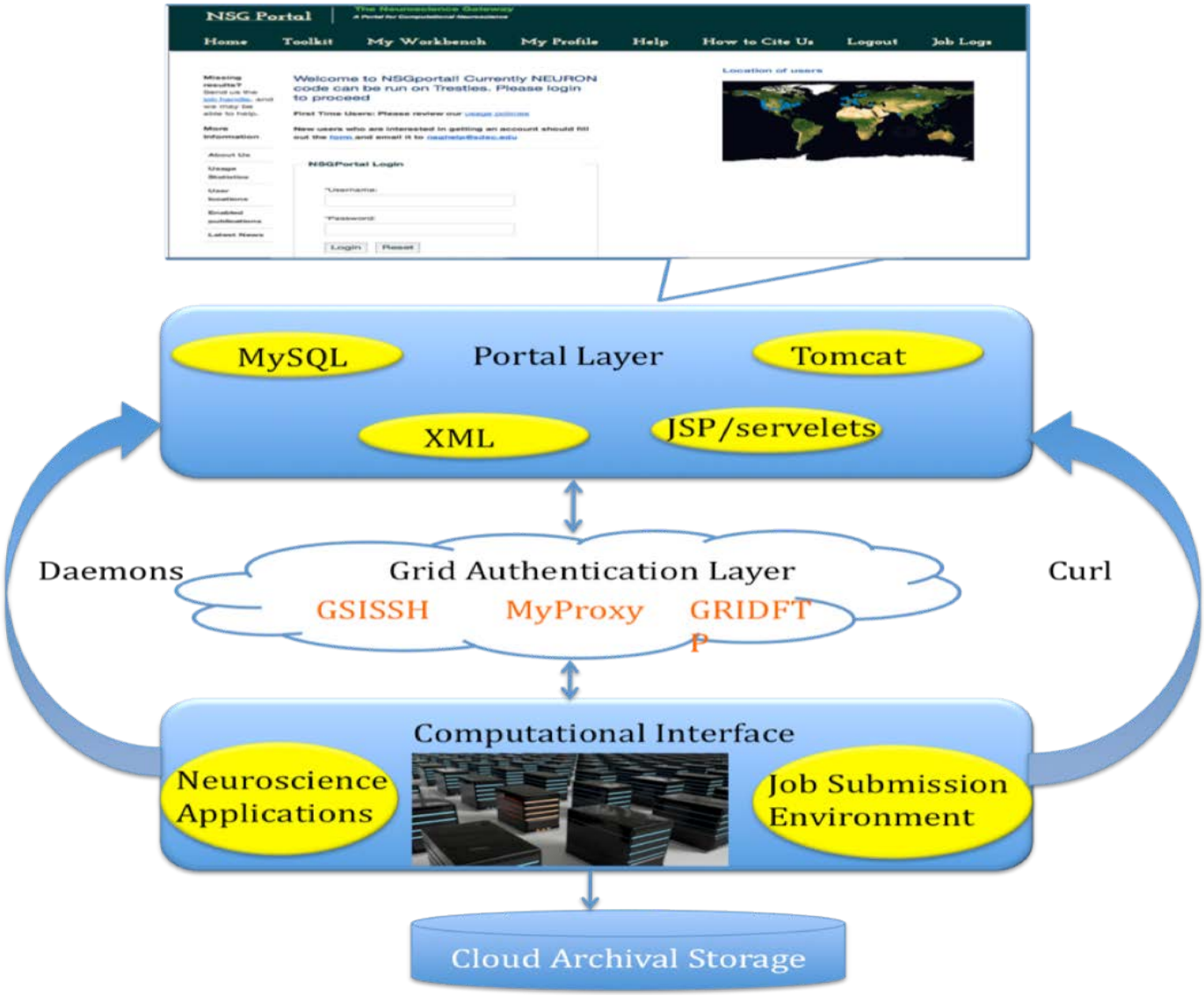


Fig. 2. Functional Diagram of NSG Architecture.

Some of the core functional implementation changes are discussed below.

Access: A web browser serves as the entry point to the NSG portal. The web browser offers a simple interface, which allows users to upload input file or neuronal models, specify neuronal code specific input files, and specify the job submission parameters such as number of cores or nodes, expected wall clock time for job completion. Users are able to monitor the status of submitted jobs and extract output files from the user-friendly portal.

Though the community gateway account is used for job submission, individual user accounts are necessary to keep track of usage and access. Some of the neuronal simulation tools, such as NEURON, require that users be able to use the Higher order calculator (hoc) [23] programming language as a scripting language for neuronal models. However due to the possibility of malicious or incorrect use or handling of hoc codes, which poses a security concern, direct user registration on the NSG is not allowed. Users are required to fill out a form with identity information, which allows NSG administrators to validate the user manually ("vetting") prior to creating their account. Once registered, NSG can track each individual user's access and usage, as well as enforce NSG specified usage policies. The account information and usage is stored in the NSG MySQL database at SDSC.

Installation of computational neuroscience tools: Currently NEURON 7.2, NEURON 7.3, PGENESIS 2.3, NEST, Brian and PyNN have been installed on SDSC's Trestles HPC machine and are being installed on TACC's (Texas Advanced

Computing Center) [24] Lonestar HPC machine. These codes are available through the NSG for the neuroscience community. We are also in the process of installing the MOOSE tool on the SDSC Trestles machine. Based on input from users, additional tools will be installed in the future.

User input file and job distribution: Most neuroscience computational models usually have more than one input file from sources such as ModelDB [25]. To accommodate this requirement, we have added capability for NSG users to upload input file in a zip format. Many other science gateways use flat text file as input and use precompiled executable to run their job. Existing WF architecture can only handle input data that is not binary. For the NSG we had to add and implement the functionality to uuencode the uploaded zip file and to uudecode the zip file on the computational resource during the staging of input. NSG allows compilation and running of user's code based on the requirement of the neuroscience application (e.g. NEURON, GENESIS 3, MOOSE, NEST, Brian and PyNN). NEURON allows custom C++ code to be used for new equations within a particular model. To accommodate this, we created a mechanism to collect all such code (located in .mod files) and compile them as a part of the job submission process. Job scripts are automatically created and submitted. Once a job completes, the working directory is compressed along with the input files, job submission scripts and output files, and are transferred to SDSC's Cloud storage. The compressed file is also made available for immediate download through the NSG portal. File staging is handled via the Java CoG Kit GridFTP API and Java runtime exec of Globus "gsissh" is used to remotely run commands. While the job is processing on the HPC cluster, an intermediate results view option is available in the portal which gives a snapshot of the working directory that was created in the backend HPC cluster. Advanced users can look at the intermediate results folder to see if their job has started or if any output file has been written. Another notable feature is the ability to clone a job on the portal. Users are able to clone their jobs, and this is helpful when they want to submit a job with the same input file but vary the parameters such as number of cores or the wall clock time. This is particularly helpful in parallel scaling performance studies on HPC machines for neuronal tools.

Storage and data retrieval: The output data is saved as a zip file and is made available on the portal. Email notification is sent to the users when a job completes. This is handled by a curl command in the job submission script, which notifies a servlet in the web application when the job finishes. In case of curl failure, two daemon processes named "checkJobsD" and "loadResultsD" check to see which jobs have finished and transfer the result to the NSG. The NSG also moves the data from the HPC resource's scratch disk space to SDSC's Cloud storage for archival storage and employs a storage policy based on data last accessed.

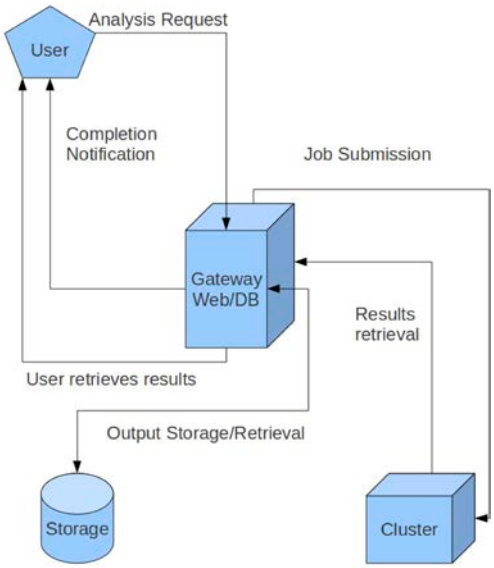


Fig. 3. User's View of NSG Flow of Work.

C. Allocation and Policies

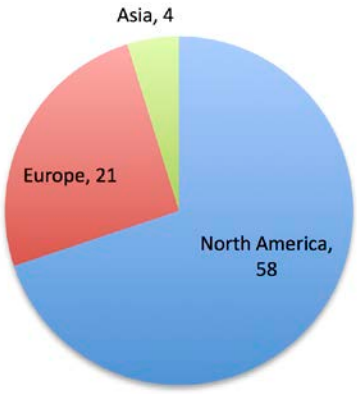
Initial allocation, called Startup allocation within the XSEDE allocation process, was obtained for 50,000 core hours on SDSC's Trestles and 50,000 core hours on TACC's Lonestar machines. Utilizing the XSEDE allocation process [XRAC] [26] we obtained community gateway account on NSF high performance computing resources, which include Trestles and Lonestar. Additional computational resources will be added based on user demands.

Users of the community gateway account abide by the policies set by the NSG administrators. Currently we allow 5000 core hours per user per year. Based on the total amount of computer time acquired every year for the NSG and the total number of NSG users, we will decide what percentage of the total time can be allocated freely to each user and monitor their usage. NSG has the capability to allow users to run jobs with their own allocation. For assessment of impact to identifiable members of the community, individual gateway user's tag will be propagated through the batch system, such that final job accounting reconciliation process will report quantitative usage by those individual gateway users. A ticketing system is in place and is used to keep track of user questions and provide immediate assistance.

D. User Workflow

Fig. 3 shows at a high level and from a neuroscience user's point of view how the flow of work will appear as a simple environment. It consists of the following steps: User logs into the NSG -> User uploads input data -> User requests simulation run on a HPC resource -> NSG frontend sends input data and job request to the remote HPC machine -> NSG retrieves output data from the remote machine -> NSG notifies

users of job (completion) status -> NSG provides user information about output data location and retrieval process.



.

Fig. 4. Distribution of Location of NSG Users.

E. Education and Collaboration

As a part of the NSG education and outreach activity, a high school student created a tutorial on Multiple Sclerosis using the NEURON code. The tutorial is now available on nsgportal website [27]. An undergraduate student from the University of California San Diego performed parallel scaling study of various models, available from ModelDB, on HPC resources located at SDSC. As a part of this study it was shown that consistency of a model was not affected by running with different number of processors [28]. Effort is also underway to make output of parallel models, from ModelDB, available for educational purposes.

F. Usage

Early users were added, following the "vetting" process, to the NSG since the beginning of December 2012. Within the first three months period we have 83 users, out of which 25 are from outside the US and this distribution is shown in Fig. 4. 45 users attended the first NSG workshop, held at SDSC in mid-March 2013 and this was simultaneously broadcasted over the web for remote attendees. The initial allocation of 50,000 core hours on Trestles was fully used up within the first two months since December, 2012 and as a result, we acquired additional 200,000 core hours for the NSG on Trestles. This demonstrates the interest and initial success of the project. From now on we will continue to write allocations proposals for XSEDE HPC resources annually and provide the computer time to NSG users. This alleviates the need for the NSG users to deal with the allocation process directly.

G. CIPRES Adaptation Experience

From the very outset of developing the NSG we decided to use the CIPRES WF as our software base. The key reasons for choosing CIPRES WF and adapting it to create the NSG are:

1. CIPRES WF is well established gateway software which has been developed over the past 10 years by experienced software developers and researchers
2. CIPRES WF has been successfully used for building gateways in other domain sciences
3. CIPRES WF developers are researchers at SDSC and as a result we were able to get expert help when needed
4. Reuse of existing NSF funded software was considered a good practice
5. Any additions or modifications done for NSG will be contributed back to the CIPRES WF software and can be adopted by future gateway developers

Bringing up the NSG to early production took about 2 months of time using approximately 75% effort of a staff person. All of the IT resources, such as VM servers, SDSC Cloud storage etc. are located at SDSC and this was beneficial to the project. Reusability of software played a key role and helped to save lot of time and effort. As a result we were able to make the portal available to computational neuroscientists within a relatively short period of time since the start of the project.

IV. FUTURE WORK

NSG portal occasionally faces issues related to using GridFTP or the NFS server on the remote HPC cluster. During file staging from the portal to the remote cluster or while copying the results back, the task would fail either because there are too many open GridFTP connections on the remote cluster or the NFS home directory on the HPC cluster is slow due to multiple users using it. Addressing this issue requires restarting autofs by system administrator of the HPC cluster. To avoid this, we are planning to use our own disk space and mount the HPC cluster's home directory on the new disk space.

We will also integrate a *programmatic interface module*, which will provide an interface layer for external neuroscience frameworks such as ModelDB, NIF [29] etc. and will allow mapping of user requests from these external frameworks to the NSG. The module will translate user requirements and authentication to the NSG interface. The external framework would format its request in XML for job processing, status query, and output data retrieval. REST API will also be incorporated to provide programmatic access to NSG. An interface for model sharing with data provenance will be provided. Users who are willing to share their models or output will be able to do so in a collaborative environment. Validated data models will be provided for educational purposes.

ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation awards DBI (Division of Biological Infrastructure) 1146949 and DBI 1146830, and by NIH awards NS011613 and DC09977. The work was also supported by computer allocation time provided by the NSF funded XSEDE organization on SDSC's Trestles and TACC's Lonestar HPC resources and by XSEDE Extended Collaborative Support

Services (ECSS) program which allowed Terri Schwartz (SDSC) to collaborate with the NSG team. The authors would like to thank Mark Miller, PI of the CIPRES software, for providing advice and guidance as the NSG was implemented. The authors would also like to thank Nancy Wilkins-Diehr (SDSC) and Suresh Marru (Indiana University) for helpful discussions, and UCSD undergraduate student Prithvi Sundar and West View High School, San Diego student Debleena Sengupta for their summer internship project work.

REFERENCES

[1] M.L. Hines, and N.T. Carnevale. "Translating network models to parallel hardware in NEURON." *J. Neurosci. Methods*, 169, pp. 425-455, 2008.

[2] http://genesis-sim.org/.

[3] http://moose.sourceforge.net/

[4] http://www.nest-initiative.uni-freiburg.de/index.php/Software:About_NEST

[5] http://neuralensemble.org/trac/PyNN/

[6] http://brainsimulator.org

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

[8] http://www.nsgportal.org

[9] R. Ananthanarayanan, S. Esser, H. D. Simon, and D. S. Modha, SC09 Proceedings, Nov 14-20, 2009, Portland, Oregon, UCA, 2009 ACM 978-1-60558-744-8/09/112009.

[10] S. Kumar, P. Heidelberger, D. Chen, and M. Hines, "Optimization of Applications with Non-blocking Neighborhood Collectives via Multi-sends on the Blue Gene/P Supercomputer," 24th IEEE International Parallel and Distributed Processing Symposium, 2010.

[11] H. Markram, "The Blue Brain Project," Nature Reviews Neuroscience 7, 153-160 (1 February 2006).

[12] http://www.nanohub.org

[13] M. Miller, W. Pfeiffer, and T. Schwartz, "Creating the CIPRES Science Gateway for Inference of large Phylognetic Trees," Gateways Computing Environments Workshop (GC), 2010, PP 1-8, New Orleans, LA, 14 Nov., 2010.

[14] https://www.gridchem.org

[15] http://robetta.bakerlab.org/

[16] http://www.xsede.org

[17] http://www.opensciencegrid.org

[18] https://neugrid4you.eu/

[19] http://bluebrain.epfl.ch/

[20] M. Miller, W. Pfeiffer, and T. Schwartz, "CIPRES Sceince Gateway: A Community Resource for Phylogenetic Analyses," TeraGrid'11, July 18-21, Salt Lake City, 2011.

[21] poplar.nics.tennessee.edu/locing!input.action

[22] http://www.ngbw.org/wbframework/

[23] Kernighan, Brian W.; Pike, Rob (1984). *The Unix Programming Environment*. Prentice Hall.

[24] www.tacc.utexas.edu

[25] http://senselab.med.yale.edu/modeldb

[26] https://www.xsede.org/allocations

[27] http://www.nsgportal.org/ed-index.html

[28] A. E. Bandrowski, S. Sivagnanam, K. Yoshimoto, V. Astakhov, A. Majumdar, "Performance of parallel neuronal models on the Triton cluster," Society for Neuroscience Annual Meeting, Washington D.C., Nov 12-16, 2011.

[29] www.neuinfo.org