Design and Implementation of a Web Portal for a NeoCortical Simulator*

Kishor K. Waikul
Dept. of Computer Science
University of Nevada
Reno, NV 89433

Lianjun Jiang
Dept. of Computer Science
University of Nevada
Reno, NV 89433

E. Courtenay Wilson
Dept. of Computer Science
University of Nevada
Reno, NV 89433

Frederick C. Harris, Jr.
Dept. of Computer Science
University of Nevada
Reno, NV 89433
fredh@cs.unr.edu

Phillip H. Goodman
Dept. of Internal Medicine
University of Nevada
Reno, NV 89433

Abstract

Over the last several years of research, we have developed a large-scale biologically realistic neocortical neural network simulator. The simulator's effectiveness as a research tool has been limited due to accessibility and ease of use. The web portal for the neocortical simulator provides online access from anywhere in the world at any time. Its GUI interface allows users to build and simulate networks in a very short period of time. This portal was built using PHP, Mysql, and a back-end running Apache on a Red Hat Linux machine.

1 Introduction

Researchers in the Brain Computation Lab at the University of Nevada have spent the last several years exploring the ways in which the brain performs computation and communication. This research was done in order to simulate correctly that computation and communication. The third version of our neocortical simulator (NCS3) is our current and most accurate simulation so far [7, 8, 9].

Although NCS3 provided the opportunity for exciting research, its use was limited due to machine access requirements and its text-based input. These limitations together with our

*This work was partially funded by the Office of Naval Research under ONR Grant xxxx

desire to increase our research collaboration led us to design the first version of a Web-based GUI portal for NCS3.

The rest of this paper is outlined as follows: In Section 2 we present the background of NCS3. In Section 3 we explain the design of our Web portal as well as its implementation. We present security issues in Section 4. Finally, we cover conclusions and future work in Section 5.

2 Background of NCS3

Currently there are two major tools being utilized by researchers to model neural activity: NEURON[2] and GENESIS[5]. These tools are used primarily to model single cells or small networks of cells in an extremely detailed manner. Because of the fine detail of activity within the single neuron, the overall network of cells in both NEURON and GENESIS is very sparsely connected. Thus, the parallel implementations of these simulations utilize a coarse-grain parallelism approach, in which one multi-compartment cell is modeled on one processor. Such an example was recently published in Neural Computation[4], where a single Purkenje cell was allocated to a single processor on a Cray T3E.

The primary goal of NCS3 is to create a novel classifier based on a biologically realistic neocortical-neural network. Parallel processing of this very large-scale, object-oriented simulator is key for approaching real-time simulation of synaptic and neocortical network dynamics. Clustering algorithms applied to the dense cell-connection matrix enable load-balancing and data parallelism by organizing highly connected groups of cells onto a particular node, thus reducing the performance cost of inter-nodal communication.

The choice for an object-oriented design for this simulator was made because the biological brain is segmented into distinct, but interrelated, parts. The object-oriented paradigm allows the simulator to model objects generically, changing their behavior through the input parameters without affecting the underlying object functionality. In this way, a user can rapidly model multiple brain regions merely by changing input parameters. It also allows the user to scale the network size more easily.

Most importantly, the object-oriented system enables us to model the relationships of neurons within a given cortical community. It is this aspect that distinguishes this model from other simulators. While other tools are excellent for modeling single-cell networks, this simulator is able to model very large scale networks of highly connected neurons. Because learning and memory occur through the relationships between active neurons, this simulator may provide a tool for developing and testing of learning algorithms of communities of cells.

The sequential implementation was finished at the end of the summer of 2000. Once this implementation was completed, it was evaluated and tested for biological realism and accuracy. This was accomplished by comparing our results with published and accepted results for channels[3], compartments[5], and synapses[1, 6].

The sequential implementation was then changed into a parallel implementation[7, 8, 9]. At this point in our research, we have tested our implementation on three architecture platforms. The first platform is a dual-processor Sun Enterprise server with 2GB of shared memory. The second platform is a Beowulf cluster of 8 Pentium II 400 machines with dual 100 Mbs Ethernet connections to a Bay network switch. The third platform is a new machine we have constructed for this project that is a a cluster with 60 Pentium III 1-Gigahertz processors, 120 GB of RAM, and a Myrinet II interconnect network.

3 Our Design

As we mentioned earlier, the motivation for this work was to provide global access for increased collaboration as well as providing an intuitive user interface. In this section we present the features the user sees from the GUI interface, the administrative tools that allow control of the portal, the database design issues, and finally the portal's communication with NCS3.

3.1 Application Features

In the design of this portal, we wished to present the user with a traditional pull-down menu interface that would provide ease of use and hopefully promote the use of NCS3. The main sections of this interface include file options, object configuration, simulation run options, and results analysis.

Files options: The integrated interface provided by our portal allows users starting from scratch to be able to design and run simulations within a few minutes. The simulations can be designed via the object configuration described later or created manually and uploaded through the file-upload option. This feature was provided to allow those users who do not wish to use the GUI design tool the ability to still use NCS3.

Because one of our primary goals is collaboration, all users can view the configuration files designed by other users; however, deletion and modification are limited to the owner of the configuration. The view/export option allows you to view the output of the simulation or to check its runtime status. The deletion of these jobs requires the owner's password (for security reasons). The logout option terminates your session with the portal and clears any non-saved session data.

Object configuration: The alternative to uploading the input setting files created manually is to use the objects menu, which provides tools for creating all the objects individually. The objects are classified into two broad categories: elemental objects, which are independent of any other objects, and hierarchical objects, which are made up of other objects. Elemental objects require a form input to set their parameters, and hierarchical objects require an intuitive graphical interface.

The form-based implementation of the reference objects stems from the assumption that most of the objects require only a slight modification of parameters in order to study a variety of real world scenarios. Another feature of the form-based objects is the online help provided for each parameter for the clear understanding of the valid input format and the significance of that parameter.

The graphical interface for the hierarchical objects allows the user to build objects out of other objects. This graphical interface is implemented in PHP and Java, backed by a Mysql database. Users of this portal can reuse any object created by any other user. This hierarchical structure proceeds all the way up to the entire brain and is illustrated in Figure 1.

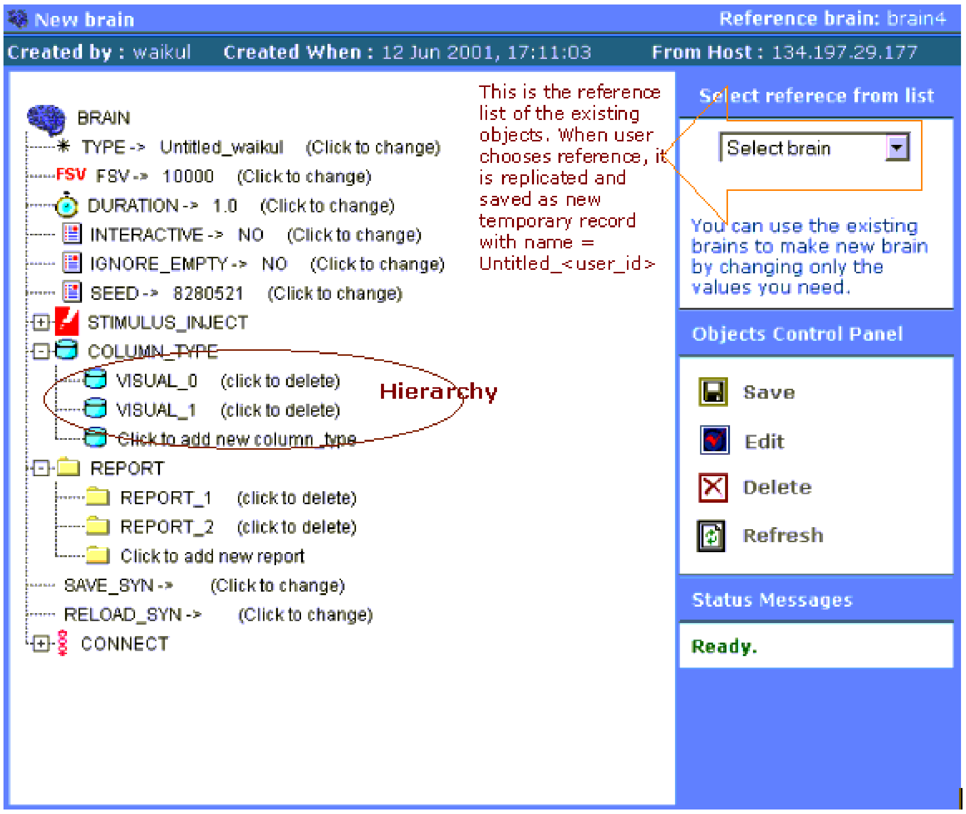

Figure 1: Object Configuration.

Simulation run options: Running a simulation requires selection of the server on which the simulation will run and selection of the brain setting input file, which is either uploaded or created as just described. The interface also provides an option for selecting the number of nodes on which the simulation is to run. Another option provides for email notification of job completion.

Result analysis: The result analysis option provides the required interface for plotting the data obtained from simulation.

When we began this research most graphical analysis was generated locally with MATLAB, but we wanted to give the user a web-based analysis tool. We implemented the results analysis with a plot program written in PHP. This program does calculation on the web server and generates a small PNG format graph, which is sent to the user. Since most of the time in the PHP program is spent on the calculation followed by the transfer of a small PNG graphics file we end up with much better performance than we would have with a Java-applet approach where the data file (which is huge) would have to be transferred to the client side and then analyzed. Another advantage is that the plotting program will also keep a cache for recently calculated data file.

3.2 Administrative Tools

In addition to the user interface, we also added administrative control via the portal. This control involves job monitoring, object formatting, and portal database administration.

Job monitoring: This part of the administrative tool-set allows monitoring of jobs and browsing all the jobs currently running. The administrator is allowed to change the priority of a job waiting in the queue. (The priority of the job is the unique number associated with the job by which all the jobs are sorted and submitted to the simulator. The highest priority job is submitted first to the simulator.) This monitoring tool also allows deleting a job from the waiting queue or deleting a running job. Figure 2 shows the interface the portal administrator would use to accomplish these tasks.

Object format edit: This option is used for changing the way an attribute is structured in the input setting file. The fields that can be edited are the content type (for example, the data type in the table), the boundary settings used to validate the parameters, the description that appears in the pop-up help, and the number of space separated values for

X	User ID	Job ID	Brain Name	Start Time	End Time	Status	Server	Results
	waikul	hercules_1	frankenstein	4 Jun 01, 15:39:43	4 Jun 01, 15:39:43	Done	gaspac	List Files
	james	last	Brain_webtest	17 Jul 01, 18:42:20	17 Jul 01, 18:43:20	Done	cortex	List Files
	james	try9	Brain_webtest	18 Jul 01, 11:24:55	18 Jul 01, 11:25:55	Done	cortex	List Files
	jiang	last	brain2cell	27 Jul 01, 15:12:00	27 Jul 01, 15:17:00	Done	cortex	List Files
	jiang	c2	MyAuditoryCortex	30 Jul 01, 12:28:53	30 Jul 01, 12:29:50	Done	cortex	List Files
	jiang	th1	MyAuditoryCortex	2 Aug 01, 4:57:09	2 Aug 01, 4:58:13	Done	cortex	List Files
	jiang	t2	MyAuditoryCortex	2 Aug 01, 19:46:36	2 Aug 01, 19:47:57	Done	cortex	List Files

Figure 2: Job Monitoring.

the parameter (if this parameter is an array). Data from this record is dynamically loaded for the validation of the parameter through the JavaScript events.

Mysql administration: We have also included in the administrative GUI interface a third-party program for administering the Mysql database. This tool is especially useful considering its ease of use and the options provided (like ordering of the columns within the table, which requires significant command line work on the database itself).

3.3 Database Design Issues

The database is a container for tables. There is a table for each of the individual objects, and the table names are constrained by some conventions to simplify the implementation of the web interface. The column names of the tables also follow conventions so that they can be easily handled by the generic code.

Other issues we have encountered include the enormous size of the result data generated by the simulator. The size of this data file is found to be megabytes. The result data was thus stored in a compressed form.

3.4 Communication Issues

We needed to take care of several issues during the design of the interface for NCS3. One important issue was how to control and monitor remotely the simulator running on clusters. The obvious concerns of this issue are the security, reliability and efficiency of the communication. Our design satisfies these requirements using the database to perform a 3-way communication.

The simulator needs to be started with a shell script that contains several parameters for different jobs. One way to reduce the information needed to be sent to the cluster is to generate the scripts dynamically on the head node of the clusters. This is done by a daemon running on the head node. This daemon's main function is to collect necessary information from the database; create the shell script for that particular job, and run the script. The database is well designed so that all the required parameters and input files for a simulation are stored in the database. The direct communication between the user and the daemon can be simplified to just a flag to tell the daemon to run the next job in the queue. In this way, there is no security and reliability issue on the direct communication between user and the daemon, and those concerns are moved to the communication between the database and the daemon, both of which should be kept on a secure network. The current implementation uses a TCP socket to send a simple "connect" message from the user to the daemon. The daemon then goes to the database to find the next job in queue and run it if the resource is available. We use Mysql for our database, which by itself provides compression, efficient management, and secure connections.

The daemon on the head node also monitors the running jobs and the system's resources. Each simulation runs in a unique directory. The daemon monitors the directory and uploads the new files generated into the database. It redirects the output of the simulator into a log file. When the simulation is done or terminated with an error, a message with proper information will be appended to the end of the log. The daemon then detects the end of the log and updates the information into the database. The user can get all the information about the simulation through the web interface, and the web page can be updated by querying the database for new information.

4 Security Issues

The overall security can be viewed in three different perspectives: application-level security, system-level security, and user-level security.

4.1 Application-Level Security

All the transactions can be encrypted by using the secure socket layer (SSL) through an https port, which requires buying an expensive certificate. This project was tested successfully with a sample test certificate provided by the OpenSSL; however, actually buying a certificate is currently not an option because of the overall scope and limited number of current users of this application. In order to prevent the attacks that typically come to a web application, the web server is separated from the simulator. The web server and the database run on the same machine. The web server and the simulator communicate through a socket discussed later. This kind of topology is called a fire-fence.

4.2 User-Level Security

This application is a trusted environment, which means the users can choose objects configured by the other users; however, they can not delete or change objects created by other users. Each user is a registered member who is authenticated by the administrator using one of the administrative options. Security at the user level is maintained by using session keys. Deletion operations require a password for each activity. User passwords are stored encrypted in the database.

4.3 System-Level Security

Attacks on web portals usually focus on the web server. In order to reduce the vulnerability of our portal we have taken several steps including removing unused associations (e.g. associations for CGI Perl scripts); removing unused programs (especially command interpreters and shells); applying security patches to the OS, web-server, and database as they are made available; and regularly checking the server logs.

5 Conclusions and Future Work

At this stage we have implemented the web portal for NCS3 as described. This has allowed an increase in collaboration in brain research and will permit us to expand that collaboration further. The option of building the brain simulation with the GUI interface or uploading your own data file has been received favorably by new, as well as experienced users. The work is far from over.

While considering the security of the web portal used in this application, the possible use of SSL (https: secure http) was investigated, implemented, and tested, but was not put into the production code. Because the expected load on the portal was thought to be limited, this security step was not given priority over the cost of the digital certificate. However, if the audience grows, moving to SSL will be necessary. SSL will encrypt and decrypt the transactions between the client and server.

Currently the kind of online help (this is not be confused with online shorthand help for objects attributes) being provided is static HTML content and any change in the format of the attribute makes necessary a series of changes in the help file (which can be tedious). This problem can be avoided by creating a Java applet based hierarchical help (similar to the objects hierarchy), which is loaded from the table containing shorthand help on objects and formatted properly using the some general format. Additional help files can be generated in similar way. This would ease the modification of the help data.

Message passing through the socket is currently minimized to a simple "connect" message to wake up the daemon. In the future, if we set the daemon to wake up automatically every several seconds, the socket communication can be completely discarded. Thus the organization will become linear rather than the current triangle relation between the web server, database, and NCS3. At that stage the jobs will be queued in the database (the job table), and the daemon will wake up to dispatch a new job when an old job is done (or after some amount of idle time). Because all the status information will be stored in the database, there is no need for direct communication between NCS3 and the web server.

References

[1] Anirudh Gupta, Yun Wang, and Henry Markram. Organizing principles for a diversity of GABAergic interneurons and synapses in the neocortex. *Science*, 287:273–278, January 14 2000.

[2] M.L. Hines and N.T. Carnevale. The NEURON simulation environment. *Neural Computation*, 9:1179–1209, 1997.

[3] Dax A. Hoffman, Jeffery C. Magee, Costa M. Colbert, and Daniel Johnston. K^+ channel regulation of signal propogation in dendrites of hippocampal pyramidal neurons. *Nature*, 387:869–875, June 26 1997. Correction in volume 390 pg 199.

[4] F.W. Howell, J. Dyhrfjeld-Johnsen, R. Maex, N. Goddard, and E. De Schutter. A large-scale model of the cerebellar cortex using pgenesis. *Neurocomputing*, 32-33:1041–1046, 2000.

[5] Christof Koch and Idan Segev. *Methods of Neuronal Modeling*. MIT Press, Cambridge, MA, 2nd edition, 1998.

[6] Walter Senn, Henry Markram, and Misha Tsodyks. An algorithm for modifying neurotransmitter release probability based on pre- and post-synaptic spike timing. *Neural Computation*, to appear. Accepted February 16, 2000.

[7] E. Courtenay Wilson. Parallel implementation of a large scale biologically realistic neocortical neural network simulator. Master's thesis, University of Nevada, Reno, Computer Science Dept., August 2001.

[8] E. Courtenay Wilson, Frederick C. Harris, Jr., and Phil Goodman. Implementation of a biologically realistic parallel neocortical-neural network simulator. In *Proceedings of the Tenth SIAM Conf. on Parallel Processing for Scientific Computing*, March 12-14 2001.

[9] E. Courtenay Wilson, Frederick C. Harris, Jr., and Phil Goodman. A large-scale biologically-realistic cortical simulator. In *Proceedings of SC2001*, November 10-16 2001.