Instructions for generating network simulations on Parallel NEURON with neuroConstruct

Padraig Gleeson

July 1, 2013

***** Note: these features are still in development! Feel free to experiment with this functionality (and send on any bug reports to p.gleeson@ucl.ac.uk) but use for your core research at your own risk! *****

1 Overview

neuroConstruct can generate code for execution on Parallel NEURON, in much the same way that it generates code for serial NEURON. Much of the functionality for this is generic and can be reused for other parallel network simulators in the future (although there are no plans to support PGENESIS, we will wait for parallel support in GENESIS 3/MOOSE).

The main prerequisite for using the parallel functionality is a successful installation of Parallel NEURON. The main set of instructions for doing this are here:

http://www.neuron.yale.edu/neuron/static/docs/help/neuron/neuron/classes/parcon.html#MPI

and a useful paper for learning about Parallel NEURON is:

http://www.springerlink.com/content/r41x6t86x6033533

One essential part of getting Parallel NEURON working is enabling ssh login to nodes (including localhost) without entering your password. Ensure this is set up for any machines you wish to test on. Although neuroConstruct can generate, execute and analyse Parallel Neuron code without requiring the user to write or even look at the code, users are very much encouraged to try writing some simple examples themselves before using this functionality in neuroConstruct.

It is recommended that you use the latest version of the neuroConstruct code from the Subversion repository. Check out the code using:

git clone https://github.com/NeuralEnsemble/neuroConstruct.git

After the first successful checkout, run ./updatenC.sh (Linux/Mac) or updatenC.bat (Windows) in the neuroConstruct directory, as there are some NeuroML files needed that reside at the NeuroML Sourceforge repository. Running the updatenC command in future (as opposed to just "svn update") will ensure a consistent set of neuroConstruct source, NeuroML files and

example models.

The Java code needs to be compiled and run with the command "ant run", so install Ant too: http://ant.apache.org. Alternatively, you will be able to compile and run neuroConstruct using "nC.bat -make" and "nC.bat" respectively (replace with nC.sh on Linux/Mac) without Ant.

Users unfamiliar with neuroConstruct should browse the included examples or try the online tutorials (http://www.neuroconstruct.org/docs/tutorials) and to to get a feel for the application.

To enable the parallel functionality of neuroConstruct, create an empty file named 'parallel' in the install directory of neuroConstruct (normally /home/username/neuroConstruct or c:\neuroConstruct). This will enable the GUI elements related to generating parallel network simulations. Why is the parallel functionality disabled by default? The parallel simulation support in neuroConstruct is still in development, and is very easy to make very large and long running but meaningless simulations with this feature. While it has been tested in our lab with a number of networks, it is currently only suitable for advanced users who are willing to work with code at the development stage.

2 Generating Parallel NEURON simulations on the local (multiprocessor) machine

Ensure Parallel NEURON is installed and tested on the machine you are running neuroConstruct from (note: while it is possible to use Parallel NEURON on Windows machines, the execution of parallel code from neuroConstruct has only been tested on Linux and Macs in this lab). With the 'parallel' file present, start the neuroConstruct GUI and create a network with a few cells and some network connections (or alternatively, use the included example Ex6_CerebellumDemo.ncx). Note that the network connections must have non zero synaptic delay, as the minimum synaptic delay is used during the NEURON simulations (all nodes can run independently for this time before they need to exchange spikes). Generate and test the simulation in serial NEURON to ensure all cells are actively spiking.

Save the project and go to tab Generate then press Edit Simulation Configurations. There will be a new drop down box present for Parallel Configurations. Here a number of host/processor combinations can be chosen from, one of which is associated with each Simulation Configuration. Note the following usage of terms:

host: a single machine with its own hostname, a number of processors and a single shared memory

processor: an individual CPU on a host

There are a number of standard Parallel Configurations to choose from including: "Local machine, serial mode with 1 host and 1 processor" and "Local machine (4p) with 1 host, 4 processors". Choose the 2 or 4 local processor option. Close the Simulation Configuration Manager GUI and generate the network. There should be a summary of the number of cells on each host/processor in the network generation output. Now go to tab Export -⟩ NEURON,

and create the network scripts in the usual way. On viewing the main project hoc file, the extra lines can be seen for parallel setup of the network (e.g. *if (isCellOnNode("CG1", 0)) ...*). Try running the simulation. Note there will be no plots/3D shape windows. A console should appear giving feedback from each of the processes running. After it has finished, the simulation should be listed when the Simulation Browser (at tab Visualisation → View Prev Sims in 3D) is opened, and cell behaviour can be replayed and plotted.

If there is a problem at this point it's best to close neuroConstruct, and go to the generatedNEURON directory of the project, and try running the runsim.sh script manually. Note: this file is generated for MPICH v1.x by default. If NEURON was compiled against another version of MPI it may be possible to get the automatic running of the script to work by placing a file named 'MPICH2' (for MPICH v2.x) or 'MPI2' for OpenMPI in the install folder beside 'parallel' and restarting neuroConstruct. For more customisation, see:
src/ucl/physiol/neuroconstruct/neuronNeuronFileManager.java (search for MPICH). Note that the hoc scripts are the same whatever version of MPI is used.

Further customisation of the Parallel Configuration options are possible by editing the file src/ucl/physiol/neuroconstruct/hpc/mpi/MpiSettings.java. For example if a Parallel Configuration is required containing hostA (with 4 procs) and hostB (with 4 procs), this can be added by creating an MpiConfiguration with 2 MpiHosts. Note hostA or hostB do not need to be the local machine, a simulation can be run in this way on these machines as long as there is passwordless ssh access to both of these and the neuroConstruct project is present on a shared filesystem whose paths are identical on these machines.

3 Generating Parallel NEURON simulations on remote (cluster) machines

It is also possible to generate parallel simulations for execution on remote machines which don't share a file system. Access to the machine without password using ssh is still required, but in this case the scripts generated by neuroConstruct are zipped up (in a tar.gz file), transferred to the remote machine by sftp, unzipped, mod files compiled and the simulation started.

There are examples of Parallel Configurations using this feature in:
src/ucl/physiol/neuroconstruct/hpc/mpi/MpiSettings.java: 'CLUSTER_4PROC' etc. A RemoteLogin object is needed for these MpiConfigurations, giving the details of the remote login host, the location to store the files, the remote copy of NEURON to use, etc. Try changing the values in the directLogin object to a machine where NEURON is installed and generate the project, followed by the NEURON code. The file runsim.sh now contains the commands to zip, transfer, unzip etc. This script can be customised by editing
src/ucl/physiol/neuroconstruct/hpc/mpi/MpiConfiguration.java, but should work as is when suitable values are used in creating the RemoteLogin object.

Once simulations are completed, they can be reloaded by opening the Simulation Browser interface for the project. Non locally run simulations which have not yet been retrieved will have 'Remote simulation' in the Date modified field. To check the status of remote sims, press Reload simulation list. This will run the pullsim.sh script in the simulations folder under simulations/, which tries to retrieve all simulation data files from the remote server. If the time.dat

file is absent (this is only written on successful completion of the simulation) then 'Remote simulation' will continue to be shown, otherwise the completion date will be entered in that field, and the simulation can be reloaded as normal. Note that as some larger simulations may take some time to retrieve, closing and reopening the Simulation Browser interface after a short while may be needed before all completed simulations show up.

4 Generating Parallel NEURON simulations for submission to a queueing/scheduling system

The above scenario is applicable when code can be executed directly on the remote machine. For many systems, code can only be run on (shared) clusters after submission of a job to a queue. This system has been tested in UCL on the LEGION cluster (with a queueing/submission system using Torque and Moab), and should be easily adaptable to other systems. neuroConstruct carries out a similar sequence of actions as above, zipping and transferring the files, but instead of running the script, calls the qsub command on a script subjob.sh, which submits the job to the queue. The completed simulations are retrieved in the same way as above.

To customise this behaviour, add an MpiConfiguration similar to legionLogin in src/ucl/physiol/neuroconstruct/hpc/mpi/MpiSettings.java. It is likely that the generated runsim.sh/subjob.sh may have to be altered by editing the script file generation in src/ucl/physiol/neuroconstruct/hpc/mpi/MpiConfiguration.java.