Running multicompartment models on a grid

- **Mike Vella**
- Department of Physiology, Development and Neuroscience, Cambridge, UK

The Project

Aim - Use a evolutionary algorithm with ~1,000,000 NEURON simulations to parameter fit multicompartment models.

A Good idea - Use Cambridge University grid (CamGrid) for required computing power

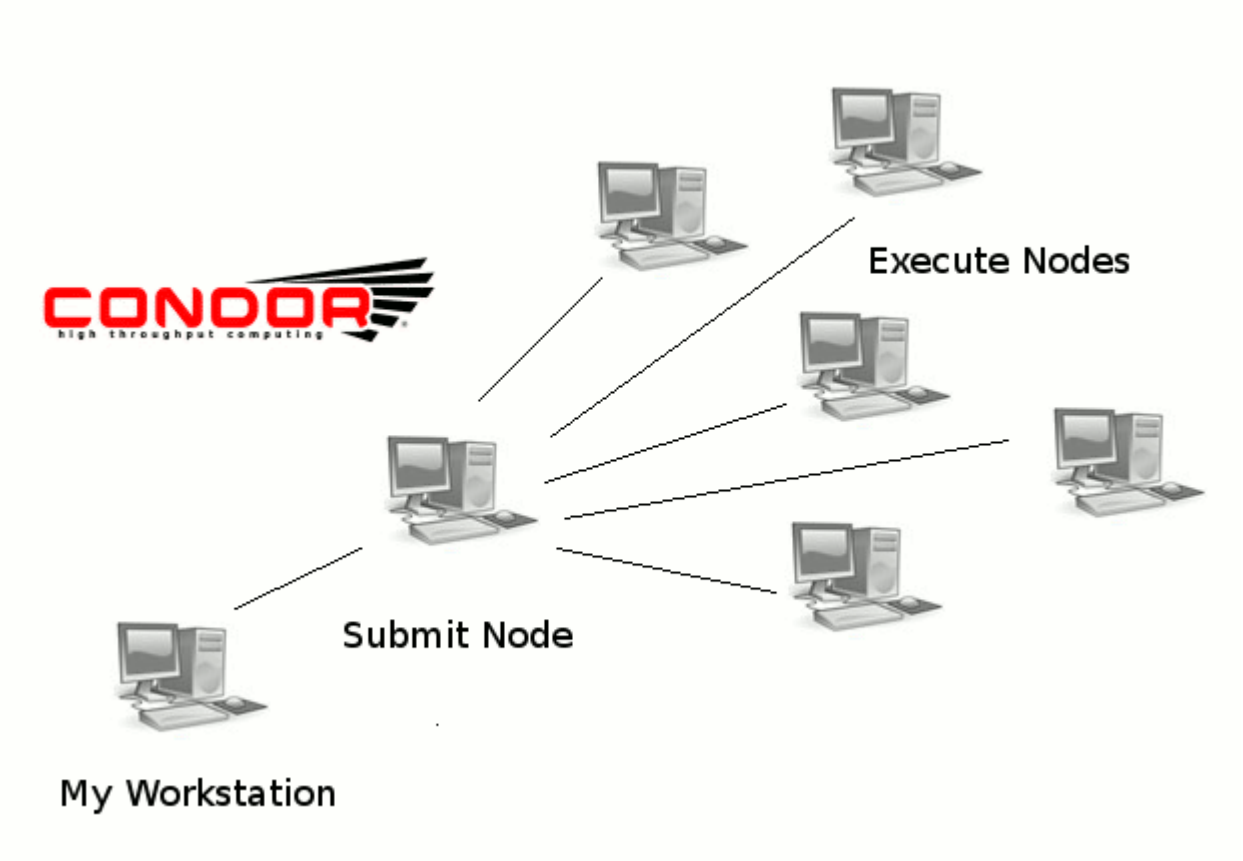
Technical Issues

Issue 1 - How to manage all this data?

Issue 2 - How to run simulations on a heterogenous group of (hundreds of) machines I have little control over and know almost nothing about

CamGrid

- Distributed computing resource based on the Condor middleware
- Run my jobs in other people's idle time (cycle scavenging)
- Many participating departments: Astrophys, Bio, Soft Systems, Chemical Informatics,
- Earth Sciences, Plant Sciences, Semiconductor Physics and many many more...



Data management: SQLite “db” API

- *File-based, embeddable* database system
- Easy to write an API specific to your problem


```
import db

writer=db.db_writer.get_db_writer(sim_var) #create writer object
#write all recordings, label appropriately:
writer.write_timeseries(t_vec,v_vec,label='voltage_mV')
writer.write_timeseries(t_vec,v_vec_2,label='soma_voltage_2_mV')

#record what input parameters were in the db:
writer.write_sim_var()

#write results of a calculation - this is a trivial example
writer.write('example value',3)
writer.write('example value_2','can also be a string')

return v_vec, t_vec
```

- Loading the data...

```
loader=db.db_loader.get_db_loader('/home/mike/dev/nrnproject/sims/output.sqlite',
                                  'sqlite',experiment_id=1)

print loader.get('simulation_name')

#example of getting an item which was result of a calculation,
#saved using writer.write method of writer object at runtime
print loader.get('example value')

#loading the voltage-vector of a recording:
soma_voltage_2=loader.get('soma_voltage_2_mV')

#get the corresponding time-vector
t=loader.get_timeseries('soma_voltage_2_mV')
```

nrnproject

- Project template and workflow for the NEURON simulator using Python
- Great introduction to using NEURON and Python together, tries to force good practice
- Plots traces and saves them
- SQLite database support via db API (previous slides)
- Source management is performed with Mercurial
- Examples of using Sphinx to document your project
- Can wrap existing hoc projects
- Many similarities to Sumatra (unnecessary redundancy?)
- https://bitbucket.org/tommctavish/nrnproject

NEURON

Portable Neuron

- NEURON binaries, script to set environment variables
- http://www.srcf.ucam.org/~mv333/wordpress/
- Solves the problem of not being able to install on remote nodes
- Hope to incorporate into pypi – allow “pip install neuron”

1. Download pNEURON here

2. Once you have downloaded it do the following:

```
tar xzf portable-neuron.tar.gz
```

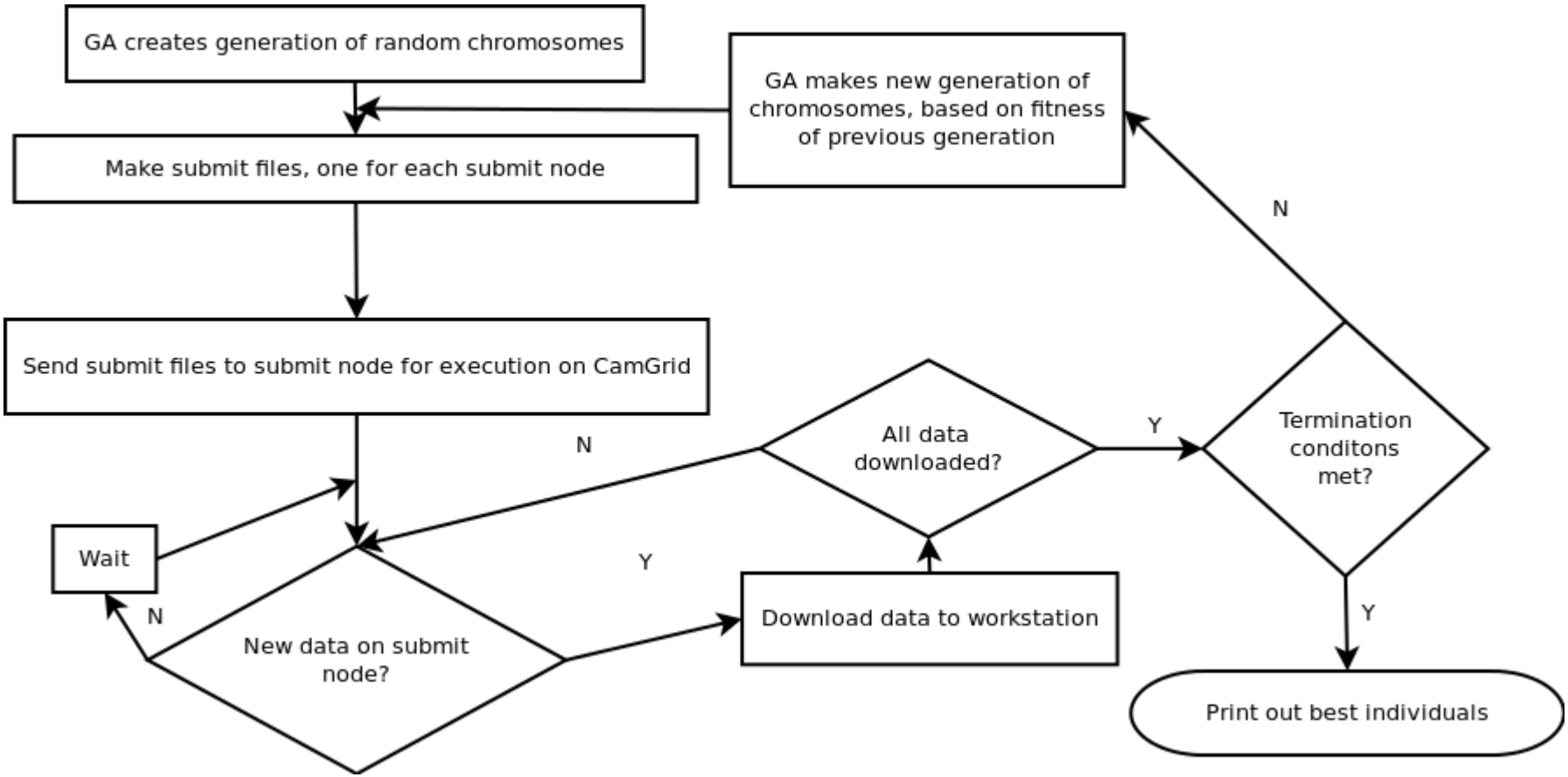
3. Then, to run python with the NEURON library you can either do:

```
./local python.sh
```

if you want to use your local Python install or

```
./pnpython.sh
```

•Workflow


GA creates generation of random chromosomes
GA makes new generation of chromosomes, based on fitness of previous generation
Make submit files, one for each submit node
Send submit files to submit node for execution on CamGrid
N
All data downloaded?
Y
Termination conditons met?
N
Y
Wait
N
New data on submit node?
Y
Download data to workstation
Print out best individuals

Summary

- SQLite “db” API useful for saving results of many simulations
- portable-neuron useful for running NEURON without compiling
- To solve a specific problem I found solutions which have probably been solved by others