PCSIM – Parallel neural Circuit SIMulator

Dejan Pecevski\textsuperscript{1},
Thomas Natschlӓger\textsuperscript{2}, Klaus Schuch\textsuperscript{1}

\textsuperscript{1}Institute for Theoretical Computer Science
Graz University of Technology
Graz, Austria

\textsuperscript{2}Software Competence Center
Hagenberg, Hagenberg, Austria
Outline

• PCSIM overview

• Bidirectional Python interface

• Network construction framework
  • Basic constructs
  • Distributed wiring algorithms
    • Supported connectivity patterns
    • Efficiency and scaling
PCSIM – Parallel neural Circuit SIMulator

• Supports distributed simulation of large spiking and analog neural networks with point neuron models.

• Implemented in C++ with a primary interface in Python
  • there is a new Java interface

• Runs under Linux, possible to port on other GNU based systems.

• Experimental support for loading NetworkML files

• Supports the standardized PyNN interface.
PCSIM – Parallel neural Circuit SIMulator Ctd.

- **Generic** network elements
  - multiple input and output, spiking and analog ports
  - suitable for hybrid simulations of spiking and analog elements, more abstract modules, neuromodulators.
Bidirectional Python Interface: Brian Network in PCSIM

class BrianCircuit(PySimObject):
    def __init__(self):
        self.registerSpikingOutputPorts(arange(4000))
        self.registerSpikingInputPorts(arange(1000))
        input = PCSIMInputNeuronGroup(1000, self)

        self.brian = brian.Network(input, P, Ce,
                                   Ci, Cinp)

    def reset(self, dt):
        ..

    def advance(self, ai):
        ..
        self.brian.update()
        self.brian.clock.tick()

net = SingleThreadNetwork()
pycirc = BrianCircuit()
pycirc_id = net.add(pycirc)

net.simulate(2.0)
Network Construction: Creating Neurons

Diagram:
- Projection
  - Connection Decision Predicate
  - Connector Factory
  - Connector Value Generator
  - Random Distribution
- Population
  - Factory
  - Value Generator
Network Construction: Creating Connections

- Projection
  - Connection Decision Predicate
  - Connector Factory
    - Connector Value Generator
    - Random Distribution
- Population
  - Factory
    - Value Generator
Distributed Wiring Algorithms

- Different types of connectivity patterns available
  - **Random** – decision whether to make a connection sampled from a Bernoulli Distribution
  - **Degree based** - the input/output degree of neurons is sampled from an arbitrary random distribution
  - **Predicate based** - independently deciding for each pair of neurons whether to connect them, probabilistically, based on their attributes

- **Example**: creating patchy long-range lateral connections of V1 neurons (Buzas et al. 2006)
  - The probability to connect neurons $i$ and $j$ is
    \[
    P(l_i, l_j, \phi_i, \phi_j) = C G(l_i, l_j) V(\phi_i, \phi_j)
    \]
    \[
    G(l_i, l_j) = e^{-\frac{|l_i-l_j|^2}{2\sigma^2}}
    \]
    \[
    V(\phi_i, \phi_j) = e^{\kappa \cos 2(\phi_i - \phi_j)}
    \]
  
  where $l_i$ and $l_j$ are the lateral coordinates,
  $\phi_i$ and $\phi_j$ are the orientation preferences of neurons $i$ and $j$ and
  $C, \kappa, \sigma$ are parameters.
Three Levels of the Wiring Algorithms

**wiring methods**
- distributed synchronized
- distributed all-to-all exchange
- simple all-to-all

**connection iterators**
- random
- predicate based
- degree distribution

**connection predicates**
- Euclidean distance predicate
- sphere predicate
- retinotopic orientation predicate
Wiring Methods

- **distributed synchronized**
  - Each node creates both its outgoing and incoming connections.
  - no MPI communication
  - A pair of nodes use the same RNG seeds when creating connections between them in order to synchronize.

- **distributed all-to-all exchange**
  - Each node creates its incoming connections.
  - Created connections are communicated through MPI.
Distributed Wiring Algorithms: Scalability

• We measured the construction time of a model where the number of used nodes increase proportionally with the number of neurons.

• In the performed experiments there are
  • 6000 neurons per node
  • on average 10000 input connections per neuron for all wiring algorithms

• Wiring methods tested:
  • Distributed Synchronized
  • Distributed All-To-All Exchange

• Wiring algorithms tested:
  • random
  • degree – each neuron has exactly 10000 input connections
  • predicate – distance dependent connection probability
Measured Construction Time

[Graphs showing the construction time for different node and neuron counts, with line graphs for different configurations: random + Synchr., random + MPI All-to-All, degree + MPI All-to-All, predicate + Synchr., predicate + MPI All-to-All.]
If you want to try out PCSIM

- The home page is: [http://www.igi.tugraz.at/pcsim](http://www.igi.tugraz.at/pcsim)
  - User manual & examples
  - Tutorial & exercises

- The source is hosted at [http://www.sourceforge.net/projects/pcsim](http://www.sourceforge.net/projects/pcsim)

- Active mailing list at Sourceforge

- Released under GNU Public License

- Publication about PCSIM