Implementation of distributed spiking neural networks with the use of message passing interface and NeMo simulator

Final Year Project

Author:
Iskander Orazbekov

Imperial College London

May 31, 2012

Abstract

Motivation behind the development of spiking neural networks arised from the growing interest in the area of neural computation and studies of brain activity in the recent years. The reason lies in the fact that SNNs incorporate time into neuron firing computation and signify the role of interconnectedness between neurons, therefore bringing the simulation level closer to reality.

However, as a result of such improvement, spiking neural networks are computationally expensive to simulate, therefore, a solution using parallel computation was needed. NeMo, a Spiking Neural Network simulator developed at Imperial College, solves this problem by making use of CUDA-interfaced GPUs with high level of parallelism.

The aim of this project is to implement the Message Passing Interface for NeMo, and by doing this, improve inter-neural communication, to allow for better performance of parallel computations. This, in turn, will allow for bigger number of neurons within the network and greater level of realism brought by the simulation.

Contents

1

Introduction

Studies in the area of neuroscience have always pushed boundaries of the development of simulation tools, requiring more computational power for realistical models. In order to achieve a high level of realism, large-scale networks are needed, that consist of more than 10^8 neurons and 10^{12} synapses - and manipulating that data alone comprises a challenging task.

Consequently, parallel computations are used, in order to minimize the workload given to a particular machine and to ensure that throughout this execution all of the clusters are communicating between each other. Once implemented, this system will allow a great amount of scalability to be put to use, enhancing the quality of simulations.

Therefore, *Message Passing Interface* (MPI) improvement in the current NeMo system will allow for faster rate of computations by making use of parallelism. With the use of MPI, it will be possible to make full use of cluster-based implementation that will deal with the scalability by distributing the workload across several machines. This means that a simulation would be able to host more neurons, therefore creating a more realistical model of brain activity and making it more useful for the research purposes.

However, bigger number of neurons results in need for more efficient memory management, as more data would have to be stored for effective communication. At the same time, a mapping specific to the topology and interconnectedness of a particular network should be derived to increase the efficiency of transmissions.

All in all, the successful implementation of the MPI communications protocol will achieve not only a greater scope of usage of NeMo simulator within neural computing research, but also provide a good basis for future implementations and enhancements of the simulator.

2

Background

2.1 Spiking Neural Networks

An *artificial neural network*(ANN) is defined as a computational model inspired by the structure and functional aspects of biological neural networks.[9] It is an adaptive system that consists of **neurons**, its basic computational units, that are connected with each other via **synapses**. Modern neural networks are mostly used as non-linear statistical data modeling tools to find patterns between inputs and outputs.[3]

Spiking neural network(SNN) - is a third generation neural network that comprises within itself not only concepts of neuronal and synaptic states, but also time.[15] As a result of putting emphasis on time as an acting property, a much higher level of realism is achieved in simulations, making SNNs particularly useful for research in such areas as brain activity or neural computation.

2.1.1 Artificial and Spiking Neurons

In order to distinguish between different generations of neural networks, one would need to take into account type of neurons, as they define the type of network.

Typical neural networks consist of a number of computational units, *artificial neurons*. An *artificial neuron* is a model of a mathematical function or an abstraction of biological neurons[8], that given several inputs, derives a value based on the sum of this collection of inputs and built-in function, and returns this value as an output, essentially acting as an axon of a real neuron. Artificial neurons differ depending on their specific model, such as McCulloch-Pitts or linear-threshold

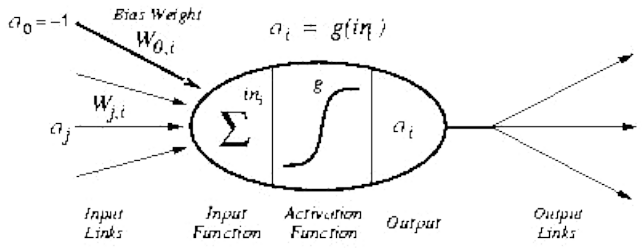

Figure 2.1: A graphical model of a simple artificial neuron[18]

function. These models define a set of properties that a particular neuron possesses, such as transfer (or activation) function.

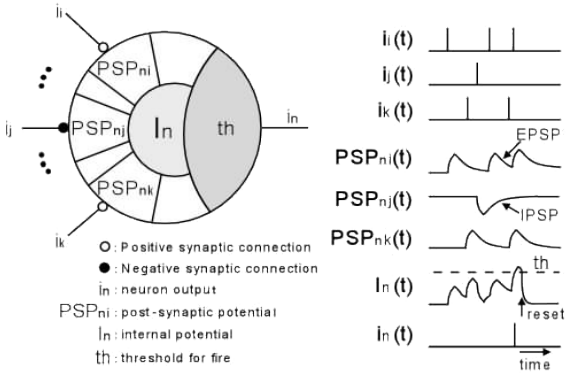

Figure 2.2: A graphical model of a spiking neuron[10]

However, spiking neural networks are quite different in nature and, specifically, in structure of their computational units from artificial neural networks. SNNs consist of **spiking neurons** that aim to model the activity of real biological neurons, that is to make an abstraction which is as close as possible to the original. A *spiking neuron* instead of having set time period of emission, fires a **spike** - a very short signal that remains at its peak value for about a millisecond. Firing in spiking neurons is caused by changes in membrane potential as well as resulting from time-based activation function. The signal is transferred to other neurons, in turn, increasing or decreasing their membrane potential. However, since signals do not vary in value, the information is transferred via a collection of spikes, called a *spike train*. Through the spike train it is relatively easy to obtain the timing and number of spikes, therefore, these two variables hold the actual data.[26] By possessing these qualities, networks built from spiking neurons obtain a significantly larger computational power than that of same-sized artificial neural network.[16]

Activation Function

An activation function is usually an abstraction that represents the firing rate of the cell, that is dependent on the sum of inputs as well as on time.[9] In spiking neural networks, this function does not output a binary set of values, but is rather calculated through a set of differential equations that depend on the input current and, if implemented, time.

Spiking Neuron Models

Here some of the spiking (or biological) neuron models will be presented.

- Integrate and Fire

  Integrate and Fire(IF) model is a simple and relatively accurate representation of the actual biological neuron. It divides the behaviour of membrane potential into two parts: long periods of *integration* and short firing of *spikes*. The activation function for this kind of neurons is a time derivative of the law of capacitance, with a refactoring time added to limit the speed of firing: $f(I) = \frac{I}{C_m V_{th} + t_{ref} I}$. Therefore, the rate of firing is directly proportional to the input current.[26]

  However, if not modified, this model does not implement time-dependent memory. Moreover, it also lacks details in representation of biological neurons, as most of the biophysiological processes are simplified. For example lagging of sodium channel activation present in Hodgkin-Huxley model is absent in IF.[12]

- Hodgkin and Huxley

  Hodgkin and Huxley(HH) model aims to incorporate an exhaustive description of a real biological neuron - therefore, requiring a large number of parameters to operate. As opposed to IF, HH model uses nonlinear differential equations in its activation function to determine the membrane charasteristics and, therefore, provide the closest biophysical representation of a biological neuron.[11]

- Izhikevich model

  Izhikevich neuron model is aiming to produce a plausible representation, close to HH, of a biological neuron, while maintaining the computational power and efficiency of an IF model. It has several improvements over IF model, such as enhancing the activation function for the purpose of more accurate spike firing representation, which is done by introducing more types of spiking periods and give the implementation.[13]

2.1.2 Synapses

Synapses in spiking neural networks represent dendrites in biological neural systems, making directed connections between spiking neurons. Therefore, synapses act as the main transmitters of spikes within the network, contributing to the connectionist approach, the main paradigm of neural networks.

Synaptic Plasticity

Synaptic plasticity is the ability of a given synapse to change the number of receptors depending on the use.[26] This represents quite an important part of the neural network model, as it is a part of real-time alterations that occur during the actual simulation.

Spiking neural networks rely on *spike timing dependent plasticity*(STDP) model for simulating plasticity, as most of information transmitted in SNNs is dependent not on the power of signals but rather on their timing and number. STDP rules divide the spikes occuring onto pre- and post-synaptic, determining the changes in the action potential that they bring in. Consequently, these parameters control the extent of synaptic modification.[20]

2.1.3 Topology

Topology of a neural network is essentially its layout, that displays the connections between the neurons. Topology plays critical role, when the neural network is mapped onto the computational clusters, as it helps distinguish an optimal way of allocating neurons between the nodes, and by doing this significantly increase the efficiency of communication.

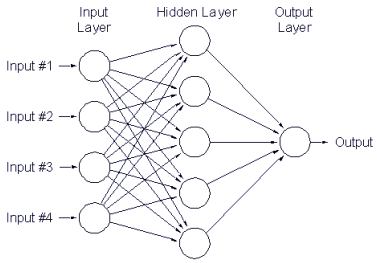

Figure 2.3: Topology of a feed-forward neural network[21]

2.2 Spiking Neural Network Simulators

In order to give a better outlook on the field of spiking neural network simulation, this section will cover some of the state of the art solutions present to date.

2.2.1 NeMo

NeMo is a spiking neural network simulator aimed at real-time simulation of large-scale neuron systems with the use of highly parallel GPUs.[1] NeMo's main purpose is to produce simulations that would be particularly useful for research, therefore, main emphasis in this tool is put onto scalability and real-time aspects of produced simulations.

NeMo is the main platform of MPI implementation for this project. The main goal is to improve inter-neural communication between computational clusters within NeMo. At the same time, neuron mapping should be changed to ensure efficient communication during the simulation.

2.2.2 Brian

Another solution, *Brian* aims for bigger flexibility and ease of use, therefore making it more suitable for teaching purposes. [6] This particular simulator will provide a good example of an spiking neural network simulator, and as it is easy to operate, will be useful for learning main concepts of SNN in detail.

2.2.3 SpikeNET

SpikeNET is a spiking neural network simulator created for large-scale integrate-and-fire networks simulations.[2] As the main aim of this project is to achieve the highest possible number of neurons hosted and computed simultaneously, this solution requires a very high level of parallelism in order to operate. Therefore, findings from this project would be useful later in the project, when the focus is going to be on the scale of computations.

2.2.4 Blue Brain Project

Initiated in cooperation between IBM and EPFL, the *Blue Brain Project* aims to produce a virtual brain in a supercomputer, Blue Gene, provided by IBM.[17]

Computational power required for the operations is immense, however, supercomputing technology gave neuroscientists a set of tools to solve this problem. The Blue Brain simulator is a very interesting project, and due to exceptionally difficult task of reducing workload across the stations - it will be particularly useful at the stage of MPI development.

2.3 Distributed Computing

As the main objective of the project is to enhance communication efficiency between neurons in the spiking neural network, use of distributed computing plays crucial role in its achieving. In this context, distributed computing means parallelization of tasks across the connected system and providing efficient communication medium between separate computational clusters.

2.3.1 Parallel Computing

Parallel computing is a form of computation where most of calculations are carried out simultaneously.[7] In order to achieve this, large problems are divided into smaller independent parts that are carried out by separate computational units, with feeding the results either into the main cluster or holding it for further computations. The rise of this particular field is due to the need in high-performance computing, especially in the light of frequency scaling becoming more and more limited because of power consumption.[14]

Due to the nature of this project, a high degree of parallelism is crucial, in order to maintain the highest possible speed of simulation - the matter is that simulation would be split between several clusters (*nodes*) which, depending on the mapping, will operate independently, after receiving the information from main node at the start.

2.3.2 Message Passing Interface

Message Passing Interface (MPI) is a language-independent protocol which acts as a communications medium for a group of processes.[25] MPI's main function is to provide a solid communication channel between the processes in a highly parallelised system, thus, enhancing efficiency of this system. Message passing programs are written in Fortran and C, with the use of built-in functions.

Currently NeMo has an MPI implementation within it, however, it is not the most optimal one. The main idea behind parallelisation in this project is that

neurons are allocated, using mapping algorithm, between several computational nodes, which are communicating with each other with the use of MPI. Therefore, one of the main objectives will be to create an MPI layer inside system, that would increase the speed of computations by enhancing quality of inter-cluster communication.

There are currently several implementations of MPI to date, so here is a brief overview of available solutions.

Open MPI

The *Open MPI Project* is an open-source implementation of MPI-2 maintained by a group of academic, research and industry partners, Open MPI Team.[19] This particular implementation aims at compatibility and high performance on all platforms, thus, making it quite easy to install and configure. Open MPI is a useful tool that will help understand the underlying concepts of message passing and generally give a good background within this subject area.

MPICH2

MPICH2 is an MPI implementation from Argonne National Laboratory, that aims at high performance and extendability.[23] The main idea of the project is to make the simulations as fast as possible, via enhancing the efficiency of communication, therefore, MPICH2 fits well with the aims set, by providing focus on the speed and effectiveness. Another reason to choose MPICH2 for this project is that this package is already installed on the lab machines, therefore, saving time for the initial setup.[24]

2.3.3 Cluster-based approach and Mapping

One of the most important part of the simulation is the initial **mapping** of neurons across the hosts, as it will affect the amount of inter-cluster communication and therefore overall efficiency of the system. *Mapping* defines the layout of the resulting system and is directly affected by topology - neurons are split into several groups depending on the number of synapses connecting those. The main idea is to keep the amount of inter-cluster communication to minimum, as it is more expensive in terms of memory and time than communication within the node.

Mapping is defined by topology and hierarchy of the system - most of implementations have a master node, accountable for adding neurons and synapses to

particular clusters, however, it is possible to have a distributed system, were all nodes are equal in resposibility and actions are taken independently.

Current implementations of cluster-based approach include:

- Izhikevich's large-scale model for simulating mammalian thalamocortical systems [5]
- IBM cortical simulator project[4]
- Neuromorphic model for GPGPU cluster by Tarek M. Taha[22]

3

Objectives

4

Structure and Design

4.1 Overview of the Project Goals

In order

4.2 Objective-specific Structure

4.3 Final Design

5

Outline of Differences

5.1 Goals

5.2 Impact on the Design

5.3 Main Improvements

6

Evaluation

6.1 Current Solutions

6.2 Criteria of Comparison

6.3 Results Collection

7

Conclusion

7.1 Outcome of the project

7.2 Possible impact

7.3 Further extensions

Bibliography

[1] Murray P. Shanahan Andreas K. Fidjeland, Etienne B. Roesch and Wayne Luk. Nemo: A platform for neural modelling of spiking neurons using gpus. *Proc. 20th IEEE International Conference on Application-specific Systems, Architectures and Processors*, Vol. 1:137 – 144, 2009.

[2] Rufin van Rullen Arnaud Delorme, Jacques Gautrais and Simon Thorpe. Spikenet: A simulator for modeling large networks of integrate and fire neurons. *Neurocomputing*, Vol. 2627:989996, 1999.

[3] Yaneer Bar-Yam. *Dynamics of Complex Systems*. Addison Wesley, 2003.

[4] Rajagopal Ananthanarayanan Dharmendra S. Modha. Anatomy of a cortical simulator. *SC '07 Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, Vol. 1:1–12, 2007.

[5] Gerald M. Edelman Eugene M. Izhikevich. Large-scale model of mammalian thalamocortical systems. *PNAS*, Vol. 105:3593–3598, 2008.

[6] D. F. Goodman and R. Brette . Front. Neuroinform. Brian: a simulator for spiking neural networks in python. *Frontiers in Neuroinformatics*, Vol. 2 Art. 5:1–10, 2008.

[7] A. Gottlieb G.S. Almasi. *Highly Parallel Computing*. Benjamin-Cummings Publishing Co., Inc., 1989.

[8] L. D. Harmon. Artificial neuron. *Science*, Vol. 129:962–963, 1959.

[9] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998.

[10] Takashi Morie Hideki Tanaka and Kazuyuki Aihara. A cmos spiking neural network circuit with symmetric/asymmetric stdp function. *IEICE Transactions on Fundamentals*, Vol E92-A:1691–98, 2009.

[11] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, Vol. 1:500–544, 1952.

[12] http://www.scholarpedia.org/article/Adaptive_exponential_integrate-and fire_model.

[13] Eugene M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, Vol. 14:1569– 1572, 2003.

[14] Vipin Kumar. *Introduction to Parallel Computing*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[15] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, Vol. 10:1659–1671, 1997.

[16] Wolfgang Maass. Computation with spiking neurons. *The handbook of brain theory and neural networks*, Vol. 1:1–14, 2003.

[17] Henry Markram. The blue brain project. *Nature Reviews Neuroscience*, Vol. 7:153–160, 2006.

[18] Online Stanford Encyclopedia of Philosophy.

[19] Timothy S. Woodall Richard L. Graham and Jeffrey M. Squyres. Open mpi: A flexible high performance mpi. *6th International Conference on Parallel Processing and Applied Mathematics*, Vol 1.:1–12, 2005.

[20] Kenneth D. Miller Sen Song and L. F. Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, Vol. 1:1–36, 2000.

[21] Michael Steinbach Tan, Pang-Ning and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, 2006.

[22] Bing Han Tarek M. Taha. Neuromorphic models on a gpgpu cluster. *Neural Networks (IJCNN), The 2010 International Joint Conference*, Vol. 1:1–8, 2010.

[23] E. Lusk W. Gropp and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press, 1999.

[24] E. Lusk W. Gropp and R. Thakur. *Using MPI-2: Advanced Features of the Message-Passing Interface*. MIT Press, 1999.

[25] Nathan Doss William Gropp, Ewing Lusk and Anthony Skjellum. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel Computing*, Volume 22:789828, 1996.

[26] Werner Kistler Wulfram Gerstner. *Spiking Neuron Models - Single Neurons, Populations, Platsticity*. Cambridge University Press, 2002.