Top-down profiling methodology and new Partition and Placement tool to improve Spiking Neural Network simulations on a massively many-core neuromorphic platform

Gianvito Urgese[1], Francesco Barchi[1], Enrico Macii[1], Andrea Acquaviva[1]

[1]Department of Control and Computer Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, IT

In this poster, we present a top-down methodology aimed at evaluating the scalability of Spiking Neural Network (SNN) simulations on a massively many-core and densely interconnected platform called SpiNNaker. On the basis of this analysis we designed a set of software tools capable to improve the reliability of the simulations and reduce the communication overhead over the inter-chip links.
SpiNNaker is a neuromorphic emerging technology that makes use of General Purpose processors to execute real-time simulations of Spiking Neural Networks (SNNs). These are networks composed by Spiking Neuron Models simulating the behaviour of biological neurons.
The system is organized in a two-dimensional toroidal-shaped triangular mesh where the SpiNNaker chips represent the processing nodes. Each node contains a router that allows the communication among the six neighbouring nodes, 128MB of shared memory and 18 ARM968 cores ensuring the parallel execution of a variety of neuronal models and synapses. Neurons constituting the emulated networks are mapped to one of the available core and their spikes, represented in the form of packets, are propagated by the on-chip router through the intra-chip network (NoC) and the inter-chip communication links.

However, due to the heterogeneity and complexity of neuron population activity, it is difficult to obtain an optimal usage of the available resources with a potential impact on the reliability of the simulations and an increasing risk of simulation crashes.
To address these challenges, we analysed the SpiNNaker simulation through a customized profiling methodology, to identify the critical behaviours of the system. Then, we focused on the Partitioning and Placement software that maps SNN on the SpiNNaker Machine with 48 chips, and finally we exploited the information extracted by means of the custom profiling methodology to design a software pipeline capable to perform a more efficient Partition and Placement.

The profiling methodology aimed at identifying those critical SNN configurations responsible for a significant loss of packets. For this purpose, we built two classes of simple SNNs generating huge flows of spikes between cores and chips. This allowed to define the most critical architectural points. Specifically, we isolated two critical points: i) In case of high level of traffic, generated at the same time by cores belonging to the same chip, some packets are lost and do not reach the router due to conflict in the access of the first level of multiplexer tree that handles the internal router inputs. A queue for packet retransmission implemented, added in some neuron models, is capable to overcome this problem. ii) High level of external traffic load passing through a single node, i.e. generating an hot-spot. To create this kind of unlikely configuration and to explore the limits of inter-chip links we generated more complex SNNs that can involve multiple chips. The execution of these configurations allowed to identify some particular problems, such as the case where packets are lost in the transmission path when one of the routers involved in the communication uses the external input ports either for input and output traffic propagation.
One possible solution to avoid this last problem, unless to generate configurations with one-way traffic that drastically reduce the available bandwidth (using three ports as input and the three as output only), consists in the reduction of exchanged packets among chips. This can be obtained

through the development of hardware-aware Partition and Placement algorithms capable to optimize the resources usage during the configuration of SNN simulations avoiding the creation of particularly dangerous neuron arrangements.
The currently used Partition and Placement software often generates, for complex networks, unstable configurations that produce unreliable results and in some cases the interruption of the simulation due to excessive traffic over the intra-chip links. This behaviour, in most of the cases, turns out to be caused by a deadlock condition generated among some of the chips involved in the transmission path.

Inspired by VLSI techniques, used to place logic gates on the SoC devices, we designed a Partition and Placement software solution based on Spectral Clustering capable to optimize the SNN placement in order to reduce the exchanged packets rate among chips.
Spectral Clustering is implemented using a traditional clustering algorithm, such as K-means, on the points of the problem identified in the space of the eigenvectors of a similarity matrix extracted from the data set. This is done in three steps: i) Starting from the directed graph of neurons it is generated the Laplacian matrix; ii) neurons are clustered in groups using the spatial information of the eigenvectors; iii) The more convenient neurons implicit positioning on the SpiNNaker nodes is extracted.
In order to automate the process of partitioning and placement we implemented these procedures in a python module called GrapH Optimizer SpiNNaker Tool (GHOST), able to run the Spectral Clustering and properly configure the SNN for simulation with SpiNNaker.
The general idea of Spectral Partitioning is to assign a weight to all the edges of graph representing the SNN taking into account both the connectivity among neurons and the estimated firing rate calculated on the basis of its simulation parameters. In this first implementation we distinguish only between neurons with low or high activity.

We tested GHOST to build the configuration files of the simulation implementing the biological plausible Cortical Microcircuit (CM) and compared the performances, in terms of traffic and successful end, with the standard Partition and Placement software.
From this analysis, two advantages can be observed: i) For some CM networks scaling factors the default network generated by standard software were interrupted due to an error of the core run-time while GHOST generated configuration works properly; ii) In simulations where both configurations are able to terminate with useful results the GHOST version presents up to 98% (for very specific cases) less packages circulating in the network.
Further tests have verified that the technique of Spectral Clustering allows to place neurons on the architecture by reducing the traffic among nodes of about 15%.