Work-in-Progress: Impact of Graph Partitioning on SNN Placement for a Multi-Core Neuromorphic Architecture

Work-in-Progress: Impact of Graph Partitioning on SNN Placement for a Multi-Core Neuromorphic Architecture

Francesco Barchi, Gianvito Urgese, Enrico Macii, Andrea Acquaviva
Politecnico di Torino
Torino, Italia
francesco.barchi@polito.it

ABSTRACT

In this paper, we evaluate a partitioning and placement technique for mapping concurrent applications over a globally asynchronous locally synchronous (GALS) multi-core architecture designed for simulating a spiking neural network (SNN) in real-time. We designed a task placement pipeline capable of analysing the network of neurons and producing a placement configuration that enables a reduction of communication between computational nodes. The neuron-to-core mapping problem has been formalised as a two phases problem: Partitioning and Placement. The Partitioning phase aims at grouping together the most connected network components, maximising the amount of self-connections within each identified group. For this purpose we used a multilevel k-way graph partitioning strategy capable of generating network-partitions. The Placement phase aims at placing groups of neurons over the chip mesh minimising the communication between computational nodes. For implementing this step, we designed and evaluate the performances of three placement variants. In the results, we point out the importance of using a partitioning algorithm for the SNN graph. We were able to achieve an increase in self-connections of 19% and an improvement of the final overall post-placement synaptic elongation of 29% using the simulated annealing placement technique, compared to 22% obtained without partitioning.

1 INTRODUCTION

Finding the best way to map processes to physical cores (PCs) is an important optimization problem with significant impact on application reliability, performance and energy consumption [8]. In this paper, we evaluate the graph partitioning impact on process graph placement for a multi-chip many-core architecture called SpiNNaker. The SpiNNaker Machine is a globally asynchronous locally synchronous (GALS) architecture and was born in the neuromorphic field. The SpiNNaker Compute node is the SpiNNaker chip, a multi-core SoC with 18 ARM Processors, 128 MB of shared RAM and a custom router [2]. The router is the main component of the system, it allows each chip to be connected with six other chips. In this way, it is possible to build a modular system of interconnected compute nodes. The building block of the SpiNNaker architecture is the Spin5 Board, a PCB that contains 48 SpiNNaker chips [3].

Spiking Neural Network (SNN) is a neural model used by neuroscientists for simulating biologically plausible brain activity. The neurons and their synapses are modelled as differential equations capable to emulate the behaviours observed in biological networks [6]. An SNN can be described as a graph where each vertex, named *Population*, contains a group of neurons sharing the same model and parameters. Each edge (*Projection*) represents the rule used to generate synaptic connections between the neurons of two *Populations*. Using this SNN description system, Van Albada et al. [9] implemented the cell-type specific *Cortical Microcircuit* (CM) model. In our work, we use the Cortical Microcircuit scaled to 5% (3 854 neurons and 716 567 synapses) for evaluating the partitioning self-connectivity and the placement synaptic elongation.

xxx, xxx, xxx
xxx. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 METHOD

Using PyNN [1], an SNN can be represented as a graph of Populations and Projections $\mathcal{G}_{\mathcal{P}}$. In the same manner, the neuromorphic architecture can be represented using an undirected-weighted graph, $\mathcal{G}_{\mathcal{T}}$. We can define the placement problem $\Pi : \mathcal{G}_{\mathcal{P}} \rightarrow \mathcal{G}_{\mathcal{T}}$ as a minimisation problem under the following constraints: i) A processor can simulate only neurons belonging to the same population. ii) Each processor can simulate a maximum number of neurons, depending on the complexity of the model.

The metric to be minimised is called *overall synaptic stretching*, it describes the deterioration of communications due to the distance between two communicating entities after being placed.

We can redefine the problem in order to fulfil the two constraints. The first step is splitting each population $\mathcal{P}^{(i)}$ into a set of partial populations (part-population) $\left\{\mathcal{P}_1^{(i)}, \mathcal{P}_2^{(i)}, \ldots, \mathcal{P}_z^{(i)}\right\}$. In this way we obtain the *part-population graph* $\mathcal{G}_{\text{pp}}$. The second step is the placement itself, assigning each part-population to a target node. We develop three strategies for placing $\mathcal{G}_{\text{pp}}$: i) Spectral Embedding: The procedure performs a spectral analysis of the graph and applies a dimension reduction obtaining a planar representation of it. ii) SCOTCH: The procedure uses the programs available in the homonym software suite for generating the placement through a Dual Recursive Bi-partitioning [7]. iii) Simulated Annealing: A well know method, used for finding a good solution of an optimisation problem [5]. From these three procedures we obtain a placement rule Π by which we can construct a *permutation vector* π and a *permutation matrix* $\mathbf{P}_\pi = (p_{ij})$ in row form. Using the $\mathcal{G}_{\text{pp}}$ adjacency matrix A and the $\mathcal{G}_{\mathcal{T}}$ distance matrix D we obtain the reordered target distance matrix $\mathbf{D}_\pi = \mathbf{P}_\pi \mathbf{D} \mathbf{P}_\pi$ and we can use it to calculate the *overall synaptic stretching* (1).

$$f : \mathbf{e}^T (\mathbf{A} \odot \mathbf{D}_\pi) \mathbf{e} = \sum_{i,j} a_{ij} * d_{ij}^{(\pi)} \tag{1}$$

Going back to the first phase, we removed the two constraints of the original placement problem transforming them into a partitioning problem. All part-populations must contain at most a number of neurons equal to the maximum number of neurons allowed to be simulated in a processor. For solving this partitioning problem $\mathcal{G}_{\mathcal{P}} \rightarrow \mathcal{G}_{\text{pp}}$, we exploded the *Population-Projection graph* into the

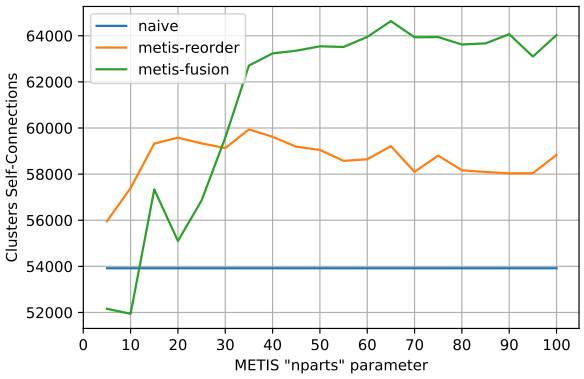

Figure 1: Exploration of the METIS parameter *nparts*. The benchmark is a $\texttt{CM}_{5\%}$ with a constraint of 200 neurons per part-population.

neuron graph $\mathcal{G}_N$ and performed a multilevel k-way graph partitioning using METIS [4]. The partitioning phase therefore takes place on the graph of the neurons in which each node is labelled according to the afferent population.

To obtain the $\mathcal{G}_{\text{pp}}$ from the clustering results of METIS we define two procedures: *METIS reordering* and *METIS fusion*. Both procedures start from the graph clustered using the METIS tools and a set of hyper-parameters. In the current state of the work, we explored the *nparts* parameter and indirectly the size of clusters identified by the tool. After the METIS clustering, each neuron is labelled with a (population, cluster) pair. Within each population the size of each cluster (intra-population cluster) is used to reorder the adjacency matrix A_N of the neuron graph $\mathcal{G}_N$. Neurons belonging to small clusters will be moved before neurons of larger clusters without mixing neurons belonging to different populations.

Starting from the reordered adjacency matrix of the neuron graph, the METIS Reordering procedure splits each population maintaining a balanced amount of neurons and the intra-population cluster-size order. In this way, the boundaries of the clusters identified by METIS are partially lost but the $\mathcal{G}_{\text{pp}}$ nodes contain the most coupled neurons. The METIS Fusion attempts to remedy the loss of boundaries of the clusters. This heuristic, at each step, identifies two intra-population clusters (belonging to the same population) whose the fusion contains a feasible number of neurons to be simulated in a processor. The selected pairs are fused together into a new intra-population cluster, this procedure continues until no more valid candidates are available. In this way, we preserve the graph cuts identified by the METIS procedure and reduce the number of part-populations.

3 RESULTS

We designed an experiment for checking the performances of three partitioning techniques: METIS reordering, METIS fusion, and a Naïve procedure that divides a population into balanced part-populations not considering network connectivity information. We used the Cortical Microcircuit model as benchmark network.

The *gpmetis* program is configured to maintain an imbalance ratio of 1.03 and a number of clusters ranging from 5 to 100. In Fig. 1 a chart depict the partitioning performance in terms of self-connections of part-populations and number of clusters. In general, the METIS related techniques leads to a good improvement over

Table 1: Impact of partitioning on four placement techniques.

Placement	Partitioning		Impact
	Naive	Fusion	
Naive	+8%	+12%	+3.23%
Spectral Embedding	+14%	+21%	+8.62%
SCOTCH	+17%	+22%	+5.83%
Simulated Annealing	+23%	+29%	+7.37%

Naïve partitioning. In particular, the METIS fusion outperforms the METIS reorder technique when clusters are small enough to be fine-grained fused into part-populations. The METIS fusion reach a self-connectivity value of 64 k compared to Naïve, which stops at 54 k, this results in an improvement of 18.5%.

In Tab. 1 we present the impact of the partitioning step on the placement performances. After partitioning, we obtained an improvement in synaptic elongation on all the considered placement techniques. Looking this data we think that the best placement performances can be obtained using the METIS Fusion Partitioning combined with the Simulated Annealing Placement. Indeed, this combination achieves a synaptic elongation improvement of about 30% compared to a random placement (median of 100 attempts).

4 CONCLUSIONS

In this paper, we evaluated the potential impact of the partitioning phase applied to the problem of SNN placement within a neuromorphic architecture. We designed a placement technique called Fusion capable of exploiting the clustering of neurons of an SNN obtained with METIS. The Fusion Partitioning creates a graph that satisfies the constraints of the architecture obtaining an improving by about 18.5% of intra-population connectivity. Finally, we evaluated the partitioning impact on the placement metric by achieving a 7.37% improvement using Simulated Annealing and 8.62% using Spectral Embedding.

ACKNOWLEDGMENT

The research leading to these results has received funding from European Union Horizon 2020 Programme [H2020/2014-20] under grant agreements no.720270 and no.785907 [HBP].

REFERENCES

[1] A.P. Davison et al. 2008. PyNN: a common interface for neuronal network simulators. *Frontiers in neuroinformatics* 2 (2008).
[2] S. Furber et al. 2006. On-chip and inter-chip networks for modeling large-scale neural systems. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*. IEEE, 4–pp.
[3] S. Furber et al. 2014. The spinnaker project. *Proc. IEEE* 102, 5 (2014), 652–665.
[4] G. Karypis and V. Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing* 20, 1 (1998), 359–392.
[5] S. Kirkpatrick et al. 1983. Optimization by simulated annealing. *science* 220, 4598 (1983), 671–680.
[6] W. Maass. 1997. Networks of spiking neurons: the third generation of neural network models. *Neural networks* 10, 9 (1997), 1659–1671.
[7] F. Pellegrini and J. Roman. 1996. Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs. In *International Conference on High-Performance Computing and Networking*. Springer, 493–498.
[8] G. Urgese et al. 2016. Optimizing network traffic for spiking neural network simulations on densely interconnected many-core neuromorphic platforms. *IEEE Transactions on Emerging Topics in Computing* (2016).
[9] S.J. Van Albada et al. 2016. Full-scale simulation of a cortical microcircuit on SpiNNaker. In *Front. Neuroinform. Conference Abstract: Neuroinformatics*, Vol. 10.