Work-in-Progress: Multiple Alignment of Packet Sequences for Efficient Communication in a Many-Core Neuromorphic System

Work-in-Progress: Multiple Alignment of Packet Sequences for Efficient Communication in a Many-Core Neuromorphic System

Gianvito Urgese, Luca Peres, Francesco Barchi, Enrico Macii and Andrea Acquaviva
Department of Control and Computer Engineering
Politecnico di Torino, Italy 10129

***Abstract*—In this era, the requirement of high-performance computing at low power cost can be met by the parallel execution of an application on a large number of programmable cores. Emerging many-core architectures provide dense interconnection fabrics leading to new communication requirements. In particular, the effective exploitation of synchronous and asynchronous channels for fast communication from/to internal cores and external devices is a key issue for these architectures. In this paper, we propose a methodology for clustering sequential commands used for configuring the parallel execution of tasks on a globally asynchronous locally synchronous multi-chip many-core neuromorphic platform. With the purpose of reducing communication costs and maximise the exploitation of the available communication bandwidth, we adapted the Multiple Sequence Alignment (MSA) algorithm for clustering the unicast streams of packets used for the configuration of each core so as to generate a coherent multicast stream that configures all cores at once. In preliminary experiments, we demonstrate how the proposed method can lead up to a 97% reduction in packet transmission thus positively affecting the overall communication cost.**

I. Introduction

Emerging many-core architectures provide synchronous and asynchronous communication layers for interconnecting the many available cores and devices. Kalray Multi-Purpose Processing Array (MPPA), Intel Lohi and SpiNNaker are three of the promising architectures described in the literature [1].

The direction taken from hardware designers is to integrate many processing elements (PE) cores and several layers of memory on each chip with custom communication infrastructures. These cores are distributed in compute clusters across the architecture, each with locally shared memory. Kalray MPPA and Intel Lohi are single-chip systems with clusters of PEs communicating using a synchronous/asynchronous infrastructure supported by a custom network-on-chip (NoC). The SpiNNaker system is a multi-chip many-core globally asynchronous locally synchronous (GALS) architecture where each computing node is a many-core SoC with 18 ARM Processors, 128 MB of shared RAM and a custom router used for addressing the asynchronous communication [2].

The router of the SpiNNaker chip is the main component of the system allowing the chip to be connected with six neighbours chips for building a modular toroidal-shaped triangular mesh of compute nodes. This router can be configured for routing both multicast and unicast packet types. The building block of the SpiNNaker architecture is a PCB that contains 48 SpiNNaker chip that is effective when used for executing problems modelled as a directed graph with a critical communication component. The board requires two preliminary phases for setting-up the applications: the task-graph placement [3] where the application is partitioned and placed on the many cores, and the configuration of cores with application-specific data-structures [4].

This last phase requires to send a list of op-codes generated in the host machine to a configurator application (pre-loaded on each core) capable to interpret these codes and create the data structures required by the final applications. Currently, the host transmits these lists of op-codes to the SpiNNaker cores by instantiating many unicast transmissions (one for each core involved in the application) even if many of the transmitted packets have the same code and could be potentially clustered in a more efficient stream. Due to physical constraints, in the communication network is vital to exploit the multicast capabilities offered by the architectures reducing the transmission of unicast packets.

In this paper, we describe a methodology that uses a multiple sequence alignment algorithm for transforming the many unicast streams, needed for configuring the system, in a consistent multicast/broadcast stream that fully exploits the features offered by the custom communication infrastructure.

II. Method

For this purpose, we clustered the op-code lists in a single sequence of packets that can be transmitted on the SpiNNaker board in a multicast style.

In the following we will show, with a simple example, how we adapted the *Multiple Sequence Alignment (MSA)* algorithm, a technique commonly used in bioinformatics, for aggregating all the sequences of op-codes in a single consensus sequence that can be transmitted with a multicast stream.

The Multiple Sequence Alignment algorithm (Fig. 1) is commonly used for aligning biological sequences such as DNA, RNA, and proteins [5] since it is capable of generating consensus alignments that preserve the original arrangement of the input sequences. These algorithms consider the four biological events (conservation, substitution, insertion and deletion) by using a scoring scheme that assigns a positive score to the match, and a negative score to mismatches and gaps found during the alignment.

In the current implementation (step A followed by step B in Fig. 2), each stream of op-codes generated from the host are transmitted to the dedicated core on the SpiNNaker board in turn with a unicast transmission. When a core is filled, the host starts filling another one. However, looking into the lists of generated streams, we recognised

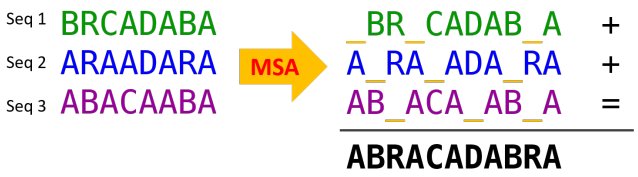

Fig. 1: Multiple alignment of three sequences of chars

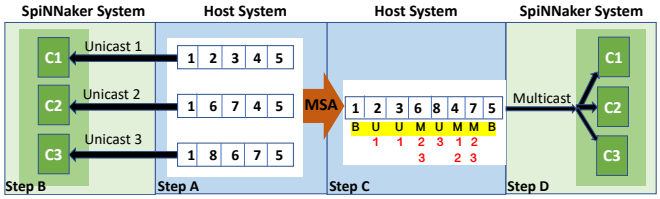

Fig. 2: Dataflow for current and MSA implementations

a high level of recursive sub-sequences of op-codes that are common for many cores. Thus, we decided to implement a pre-processing step (orange arrow labelled with *MSA* in Fig. 2)capable of clustering the recurrent pieces into a single stream that can be transmitted by means of a multicast transmission. Now packets are labelled as *Multicast (M)*, *Broadcast (B)* or *Unicast (U)* and sent to the target group of cores depending on the label.

The Clustering step is divided into four phases: **i) Encoding**, where we scan all the streams looking for recurrent op-codes appearing on multiple streams. These frequent op-codes are encoded with an integer that is then used in the MSA step, while op-codes appearing only once are encoded with the 0 value since not informative for the MSA algorithm. This last procedure reduces the effort required to the MSA that otherwise should try to align unique op-codes that will anyway be transmitted as unicast packets (Fig. 2, Step A).

ii) Alignment, where encoded op-codes are treated as input sequences for the MSA algorithm that began the computation of all the pairwise alignments, where all the sequences are aligned in pairs. Then, the score of pairwise alignments are sorted, and the best pairwise alignment is selected as a seed for the generation of the multiple alignment. The MSA algorithm, at this point, aggregates in turn the other sequences by including matches for the pieces of sequences that share the same op-codes and gaps when the sequences do not share sub-sequences. At last, MSA produces the consensus alignment that preserves the sequential flow of all the input list of packets. We configured the MSA parameters for avoiding the mismatch detection allowed by the original algorithm. The MSA algorithm can be configured with four main parameters: the *Match* value that we set to +5, a *Mismatch* score equal to -10, and two *Gap* values for considering the gap event and the elongation that we set both to 0 since we prefer to have a gap instead of a mismatch. We used an MSA function implemented in the *SeqAn* C++ bioinformatics library [6].

iii) Group definition, where we analyse the consensus alignment for defining the groups of cores that will be targeted for each op-code during the multicast transmission. In this step, we generate the multicast groups that are encoded in the routing keys.

iv) Alignment Collapsing, where the many alignment sequences are collapsed into a single consensus sequence of op-codes. The op-codes filtered from the original sequences, because the ones appearing only once, are included in the positions coherently with the sequential arrangement, and labelled as unicast for a specific core. A label with the correspondent multicast group is assigned to all the other packets (red numbers in Fig. 2, Step C).

III. Results

The figure 3 presents the number of packets generated for four versions of the Thalamo-Cortical Microcircuit (TCM). Blue bars represent the number of packets sent using unicast streams, without op-codes alignment, in which data is transmitted as it is (steps A and B in figure 2 show the flow). Green/red/yellow stacked bars show the number of packets sent using a multicast stream generated with the MSA procedure designed for clustering recurrent packets (steps A, C and D in figure 2).

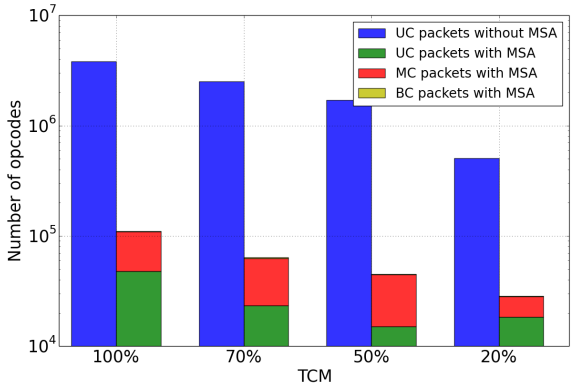

Fig. 3: Packets comparison

TCM	Chip	Core	Saved
100%	157	390	97.1%
70%	70	274	97.5%
50%	40	196	97.3%
20%	17	80	94.3%

TABLE I: Execution stats

It can be noticed that, as reported in table I, the number of saved packets is above the 94%. The table also presents the number of cores and chips on the SpiNNaker board allocated for executing the four scaled versions of the cortical microcircuit application.

IV. Conclusions

In conclusion, we developed a system able to cluster the information in order to exploit the multicast network, reducing the number of packets generated of a quantity up to the 97% and improving the host-board communication phase. The described procedure, in principle, can be applied to all type of packet transmissions towards the communication mesh for reducing the bottleneck given by the data transmission phase.

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] Wikipedia. (2018) Ai accelerator. [Online]. Available: https://en.wikipedia.org/wiki/AI_accelerator

[2] S. B. Furber *et al.*, "Overview of the spinnaker system architecture," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2454–2467, 2013.

[3] G. Urgese *et al.*, "Optimizing network traffic for spiking neural network simulations on densely interconnected many-core neuromorphic platforms," *IEEE Transactions on Emerging Topics in Computing*, 2016.

[4] A. Siino *et al.*, "Data and commands communication protocol for neuromorphic platform configuration," in *Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 2016 IEEE 10th International Symposium on*. IEEE, 2016, pp. 23–30.

[5] C. Notredame *et al.*, "T-coffee: a novel method for fast and accurate multiple sequence alignment1," *Journal of molecular biology*, vol. 302, no. 1, pp. 205–217, 2000.

[6] K. Reinert *et al.*, "The seqan c++ template library for efficient sequence analysis: a resource for programmers," *Journal of biotechnology*, vol. 261, pp. 157–168, 2017.