In-Network Online Data Analytics with FPGAs

Ryan Cooke, Suhaib A. Fahmy
School of Engineering
University of Warwick, Coventry, UK
Email: ryan.cooke@warwick.ac.uk

I. Introduction

The growth of sensor technology, communication systems and computation have led to vast quantities of data being available for relevant parties to utilise. Applications such as the monitoring and analysis of industrial equipment, smart surveillance, and fraud detection rely on the 'real-time' analysis of time sensitive data gathered from distributed sources. A variety of processing tasks, such as filtering, aggregation, machine learning algorithms, or other transformations to be carried out on this data in order to extract value from it. Centralised computation strategies are often deployed in these scenarios, with the majority of the data being forwarded though the network to a datacenter environment, typically due to the lack of required computational or storage resources at the leaves of the network, and data from other sources or historical data being required. This approach has also traditionally been viewed as more scalable, as resources can be augmented through the addition of extra compute hardware and cloud services.

However as the amount of available and potentially useful data to be processed grows, sending it all to a central location leads to impractical latencies, bandwidth consumption, and resource requirements. An alternative strategy is to deploy 'in-network' computation—performing some of the processing tasks on the data as it moves through the network infrastructure towards the datacenter. While performing processing at the very edge of the network may not be possible due to resource constraints at these nodes, many network elements such as routers, switches and gateways that act as intermediate nodes utilize more capable processors and FPGAs to perform their required functions and allow for vendor updates post deployment. These devices can be extended to offer distributed computation support for such applications.

Carrying out these tasks on the data in transit using compute resources placed at network nodes as they move towards their destination has several advantages. These tasks often reduce the amount of data, producing some form of aggregated or filtered results, thus reducing bandwidth requirements. Additionally, tasks can be carried out in parallel at different locations, and closer to the data source in a distributed fashion, potentially reducing latencies due to data queuing and communication delays. Making use of FPGAs deployed in the network infrastructure also enables support for the utilization of accelerator architectures that can boost performance for computationally intensive processing tasks.

II. FPGAs for In-Network Computation

Processing tasks that are common in real time analytics applications, such as complex event processing, aggregation, filtering, and machine learning classifiers can often be computationally intensive. Embedded software processing within the network would introduce unacceptable delays, and hence, these tasks would have to be carried out on more capable hardware closer to the datacenter with similar bandwidth costs. Hardware implementations of these tasks on FPGAs offer substantial reductions in computation time and greater performance per watt than CPU and GPU alternatives [1], [2]. Deploying these accelerator architectures on FPGAs such as those already present in the network infrastructure (currently utilised for for switching and packet processing) can allow these tasks to be pushed further down towards the edge of the network, reducing bandwidth. By opting for this embedded processing approach, it also avoids the cost and complexity of deploying 'cloudlet' servers that offer datacenter compute elsewhere in the network.

This approach to in-network analytics requires a virtualised approach to accelerator deployment, that leverages advances in partial reconfiguration (PR) [3]. This can allow multiple processing tasks to process data independently, and be modified and switched out without affecting other tasks. An approach where larger FPGAs are placed in switches with a static region managing the networking tasks and virtualised slots enabling compute-intensive processing is envisaged. This flexibility is essential due to the constantly evolving nature of large data analytics applications.

III. Current Work

We have developed a mixed integer linear programming (MILP) model to explore the effects of placing compute and different hardware platforms at nodes in a network. Existing models are mostly from the sensor network space and are typically application specific [4] assuming fixed resources at nodes [5]. They are also focused mainly on one objective, typically end-to-end latency. They are focused on distributing software tasks to processors, while in our scenario a mix of software and hardware implementations of tasks could be selected.

Given user defined constraints, the placement of compute tasks can be optimised to achieve the lowest latency, financial cost, bandwidth requirements, or energy consumption, with constraints on the other metrics not selected. Both the set of network nodes and tasks to be carried out form directed

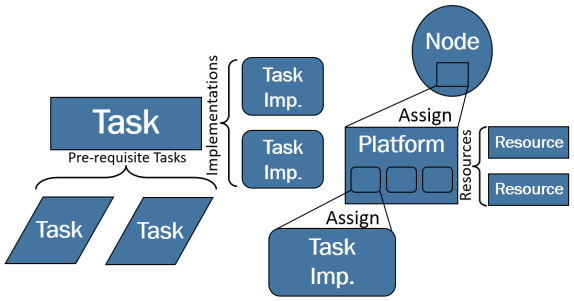

Fig. 1. Diagram of modelling - hardware platforms, and implementations of tasks are assigned to nodes

acyclic graphs (DAGs). Each task must be assigned to a network node through an 'implementation'. Each task can have several possible implementations with different execution times and hardware resources required. Tasks reduce the data they operate on by a specified factor. Nodes are also assigned a hardware platform, for example different FPGAs, processors or, microcontrollers each with a set of resources that can be used to implement tasks, a power consumption, and financial cost. The MILP takes into account all the necessary constraints to present a feasible mapping, such as task dependencies, resource requirements and data availability.

IV. Example Applications

An example of a relevant scenario is an autonomous drone disaster management system [6]. A network of drones capture images or video of an affected area with the goal of locating survivors, using computer vision techniques. They transmit the images through other drones to cluster heads in close proximity to a base station, which then forwards it to a server to perform the analysis. Transmitting hi-resolution images or video is inefficient however, so performing more processing at the drones is desirable. Other scenarios include smart traffic

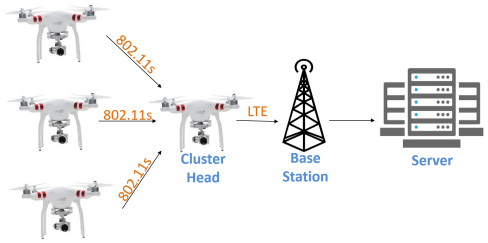

Fig. 2. Example application - drones collaborate to survey disaster struck area for survivors.

light systems, smart grid wide area monitoring, and automated surveillance on-campus IP camera networks. Our model is at the stage where it can be applied to these examples.

V. Future Work

In the near future we plan to construct example applications, such as the drone example, and experiment with the placement of compute in the network. Results can be used to validate the model detailed in Section III, and be used with the model to investigate what happens when the network is scaled up. These experiments can also be used to inform the direction of the next part of the project.

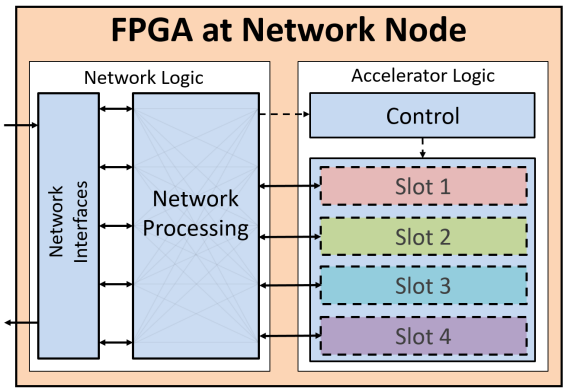

Fig. 3. FPGA infrastructure to assist with deployment of accelerator architectures onto network nodes

This will lead to the development of an FPGA framework to support deployment of virtualised accelerators onto FPGAs that are already used in the network infrastructure using PR (Figure 3). We will build on previous work with ZyCAP [7]. The framework will partition spare FPGA resources into virtual slots and manage the allocation of accelerators to these slots seamlessly with control over the same network interfaces. Data can be passed to the appropriate task slot if it is available or the required task logic can be loaded into an unused slot. The low overhead of deploying FPGAs in such a scenario is key to the feasibility of this idea, and hence the management framework must be capable enough to abstract low level operations. Unlike similar works, we aim for a fully virtualised approach that integrates compute logic onto FPGAs already used for networking functions.

References

[1] M. Papadonikolakis and C. Bouganis, "A Novel FPGA-based SVM Classifier," in *International Conference on Field-Programmable Technology*, 2010, pp. 2–5.

[2] B. Van Essen, C. Macaraeg, M. Gokhale, and R. Prenger, "Accelerating a random forest classifier: Multi-core, GP-GPU, or FPGA?" in *Proceedings of the 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines, FCCM 2012*, 2012, pp. 232–239.

[3] S. A. Fahmy, K. Vipin, and S. Shreejith, "Virtualized FPGA Accelerators for Efficient Cloud Computing," in *International Conference on Cloud Computing Technology and Science Virtualized*, 2015, pp. 430–435.

[4] M. Zeller and C. Prehofer, "Modeling and efficient solving of extra-functional properties for adaptation in networked embedded real-time systems," *Journal of Systems Architecture*, vol. 59, no. 10 PART C, pp. 1067–1082, 2013.

[5] M. Younis, K. Akkaya, and A. Kunjithapatham, "Optimization of task allocation in a cluster-based sensor network," in *Proceedings - IEEE Symposium on Computers and Communications*, no. 410, 2003, pp. 329–334.

[6] M. Quaritsch, K. Kruggl, D. Wischounig-Strucl, S. Bhattacharya, M. Shah, and B. Rinner, "Networked UAVs as aerial sensor network for disaster management applications," *Elektrotechnik und Informationstechnik*, vol. 127, no. 3, pp. 56–63, 2010.

[7] K. Vipin and S. A. Fahmy, "Zycap: Efficient partial reconfiguration management on the xilinx zynq," *IEEE Embedded Systems Letters*, vol. 6, no. 3, pp. 41–44, 2014.