Automatic Generation of Sensor Hardware by a Declarative Distributed Stream Processing System

Sheryl Coe*, Andrey Mokhov, Paul Watson
Newcastle University, Newcastle-Upon-Tyne, UK
*s.coe1@newcastle.ac.uk

***Abstract*— Stream processing is used in many different applications from wearable sensor devices and industrial Internet of Things to social networks. We propose a declarative distributed stream processing system where system indicators are monitored and evaluated to automatically generate and deploy system components to the most effective parts of the overall architecture. The system architecture comprises sensors, field gateways and cloud components. In this paper, we focus on the automatic generation and continuous update of sensor hardware during the run time of the system to enable optimization of non-functional requirements such as sensor power consumption.**

Keywords—FPGA; stream processing; declarative distributed system; Internet of Things (IoT); functional programming

I. INTRODUCTION AND MOTIVATION

The main goal of this paper is to propose and discuss a declarative and distributed stream processing system that allows the designer (i) to specify functional and non-functional requirements for an Internet of Things application as a high-level dataflow stream computation, (ii) compile the specification into a set of hardware and software components, and (iii) optimally distribute the compiled components across the available sensing and computation infrastructure, adapting the distribution in the run time, aiming to continually meet functional requirements and optimize non-functional requirements.

Designing and managing such a system can be a real challenge and understanding the constraints or benefits of where to place a system component for optimal use of resources can be complex, requiring analysis of current/past performance and the potential for future improvement across the whole system. Sensor research often cites the need for greater improvements of sensor power consumption [1, 2, 4]. By creating a system which can automatically generate the distribution of run-time components across the architecture, the system as a whole can be optimized to meet non-functional requirements such as power consumption, processing speed etc. It is intended that key system values will be monitored and evaluated so decisions on the deployment of components can evolve based on real time usage. Two approaches for this challenge are being explored, one is using continuous query language [5] based on relational algebra and the other is using functional programming language based on lambda calculus.

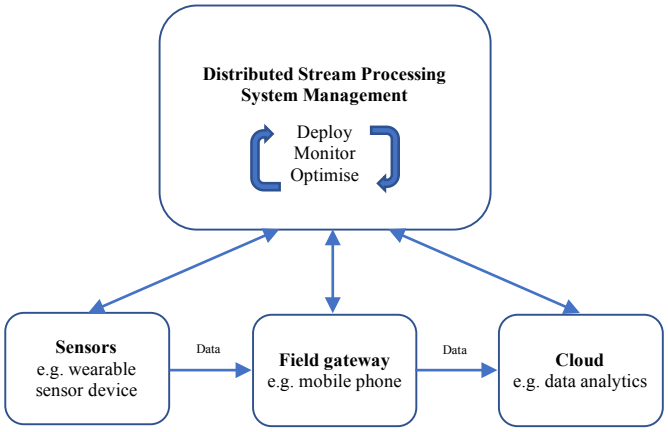

Fig. 1. Distributed Stream Processing System Architecture

One of the key questions in this research is finding the optimum allocation of the stream processing stages. Recent research has shown that moving computation between the different parts of the whole system can improve sensor power consumption [5, 3]. For example, in [5] it has been shown that sensor device battery consumption can be improved by over 400% if certain parts of the data processing are moved to the sensor device because of the resultant reduction in data transfer to the field gateway or cloud. Functional programming helps to significantly simplify the semantics and implementation of the system, because it allows us to treat computational components as pure functions that can be easily migrated from one part of the system to another by serializing function closures for transmission across the network, as described in detail in [8].

Our current research is focused on the sensor infrastructure as a part of the whole stream processing system, in particular the automatic generation of sensor hardware from the declarative system specification. Recent research has used programmable hardware to accelerate processing and reduce power consumption [1, 6, 7], however this proposal differs in that the hardware will be automatically generated from the high level system definition and will have the ability to be reconfigured during run time.

The system will be designed and built using functional programming techniques and will use the Haskell programming

language for the high-level requirement definitions and system components. It is hoped that using a functional programming language will assist with defining a formal semantics of computational components, simplifying the system verification, and will help achieve a clear separation of requirements into functional and non-functional categories, which in turn will enable a more efficient system when distributing and adapting the workload across the full system architecture – sensor, field gateway and cloud.

II. Data Streaming Architecture

The proposed distributed stream processing system includes software and hardware components spread across sensors, field gateways and cloud architecture (Fig. 1). The sensors are the beginning of the whole process logging the required events, for example temperature within an office building. Field gateways are intermediary devices used for transferring and processing sensor data, for example a mobile phone in close proximity to a sensor can stream events from the sensor using a Bluetooth connection, perform some processing if necessary and then transfer data to the cloud using the phone's internet connection. The cloud architecture is used to enable high data volume processing and analytics by making use of parallel capabilities in cloud processing systems. When the system is in use there may be a collection of sensors, constantly generating data which needs to go through multiple processing stages. Each processing stage has potentially different requirements for computational resources and could be executed anywhere in the sensor-gateway-cloud system. The distributed stream processing system will monitor, optimize and automatically deploy system components across the available architecture.

III. Moving Computation Closer to Hardware

This research is proposing to exploit FPGA (field programmable gate array) technology for wearable sensor devices with the aim of improving non-functional requirements such as power consumption. FPGAs are configured by using a hardware description language (HDL) and this research will investigate the flow of high level requirements, initially specified in Haskell, through to the HDL and hardware synthesis itself. It is proposed that the hardware will be automatically generated by the system and making use of FPGA technology will open the door for manipulating and controlling the processing power of the sensor in real or near to real time. We have developed a prototype for translating computational graphs described in Haskell into VHDL [9] and are working towards integrating it with sensor hardware and the rest of the architecture.

Another direction of research is the exploration of the advantages and disadvantages of using asynchronous processing of sensor events as opposed to synchronous processing found in the majority of FPGA components manufactured today. As has recently been demonstrated in [10], asynchronous circuits can be particularly beneficial when combined with analog and mixed-signal sensors producing data at unpredictable rates, because asynchronous circuits are event-driven, i.e. they react to changes in a system at the rate they occur, and are therefore more energy efficient compared to sensors clocked at fixed frequency. This could lead to a custom sensor device as a proof of concept for this type of implementation.

The developed technology and system infrastructure will be evaluated through its application in a range of problem domains, such as e-health, industrial IoT and smart cities. This will include a practical application of a wearable medical device for activity monitoring, where non-functional requirements will be measured and evaluated.

References

[1] Ahola, T., Korpinen, P., Rakkola, J., Rämö, T., Salminen, J., & Savolainen, J. (2007). "Wearable FPGA based wireless sensor platform." Conference Proceedings :Annual International Conference of the IEEE Engineering in Medicine and Biology Society., 2007, 2288-91.

[2] Gao, W., Nyein, H., Shahpar, Z., Tai, L., Wu, E., Bariya, M., Ota, H., Fahad, H., Chen, K. &Javey, A. (2016) "Wearable Sweat Biosensors", IEEE IEDM Technical Digest, 6.6.1-6.6.2

[3] Kalantarian, H., Sideris, C., Mortazavi, B., Alshurafa, N., & Sarrafzadeh, M. (2017). "Dynamic Computation Offloading for Low-Power Wearable Health Monitoring Systems." Biomedical Engineering, IEEE Transactions on, 64(3), 621-628

[4] Kalantarian, H., Alshurafa, N., Pourhomayoun, M., & Sarrafzadeh, M. (2015). "Power optimization for wearable devices." Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on, 568-573.

[5] Michalák P, Watson P. "PATH2iot: A Holistic, Distributed Stream Processing System." In: 9th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2017). 2017, Hong Kong: IEEE

[6] Tsoutsouras, V., Xydis, S., Soudris, D., & Lymperopoulos, L. (2015). "Swan-icare project: On the efficiency of FPGAs emulating wearable medical devices for wound management and monitoring." Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9040, 499-510.

[7] Wöhrle, H., Tabie, M., Kim, S., Kirchner, F., & Kirchner, E. (2017). "A Hybrid FPGA-Based System for EEG- and EMG-Based Online Movement Prediction." Sensors (Basel, Switzerland), 17(7), Sensors (Basel, Switzerland), 03 July 2017, Vol.17(7)

[8] Epstein J, Black AP, Peyton-Jones S. "Towards Haskell in the Cloud". In ACM SIGPLAN Notices 2011 (Vol. 46, No. 12, pp. 118-129).

[9] Mokhov A, de Gennaro A, Tarawneh G, Wray J, Lukyanov G, Mileiko S, Scott J, Yakovlev A, Brown A. "Language and Hardware Acceleration Backend for Graph Processing". Forum on specification & Design Languages (FDL 2017).

[10] Sokolov D, Dubikhin V, Khomenko V, Lloyd D, Mokhov A, Yakovlev A. "Benefits of asynchronous control for analog electronics: Multiphase buck case study". In Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017 (pp. 1751-1756). IEEE.