Towards Efficient Stream Parallelism for Embedded Devices

Renato B. Hoffmann, Dalvan Griebler, Luiz G. Fernandes

[1] School of Technology, Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brasil

renato.hoffmann@edu.pucrs.com,(dalvan.griebler,luiz.fernandes)@pucrs.br

Abstract. *Stream processing applications process raw data-flows to reveal insightful information. Efficiently coordinating the requirements of these applications is a challenge. We propose investigating high-level software solutions for these applications to achieve efficiency and high performance for embedded devices.*

1. Introduction

Stream processing applications handle information arriving in a data-flow manner, often in high speed and with real-time processing constraints [Andrade 2014]. Many applications are designed with this paradigm, where raw data streams are processed to reveal the information they might contain and report it back to the final user as insights, statistics, or suggested course of action. Meeting strict requirements for these applications requires efficient software solutions for handling and fully exploiting the underlying target hardware. Therefore, parallel programming is essential. However, parallel programming may be a challenge to the application developers. It involves low-level, error-prone, and system or architectural dependent concepts.

Traditionally, robust stream processing applications may have IoT (Internet of Things) devices simply collecting data and eventually forwarding it to a Cloud or Cluster environment where it performs the bulk of the processing. With the advent of Edge or Fog computing, computation is brought closer to the data sources for reducing the costs of data transfer [Lin 2021]. Similarly, mobile or embedded devices may also host simple stream processing applications. These scenarios share the need for integrating stream technologies in a limited-resource hardware environment.

Hardware for mobile, edge, or embedded devices is typically lower-cost (compared to regular systems) and has strict energy efficiency requirements. Additionally, this hardware is heterogeneous in nature. One possible configuration may have a Quad-core ARM microprocessor CPU, embedded GPGPU module, and 2 to 8 GB of RAM. In particular, the Nvidia Jetson board series has an interesting architectural detail: RAM shared between CPU and GPU. All-in-all, efficiently managing the resources of these systems is a challenge for stream processing developers interested in running their applications closer to the data sources or the user [Aldegheri 2018]. Providing high-level abstractions can simplify this process.

Our goal is to investigate high-level abstractions for efficiently handling parallelism and architectural details of limited-resource or embedded hardware in the context of stream processing. In particular, we want to examine means of simplifying development or abstracting the natural complexity of these systems while delivering performance (throughput and latency). For that, we propose to use a domain-specific language and

compile system. The general idea is that the developer must apply a high-level language that the compiler then uses to perform source-to-source code transformations to generate the effective parallel code. This work will be part of the author's master thesis in 2022.

2. Research Proposition

The initial steps involve investigating the literature and state-of-the-art for finding current APIs and solutions in the context of embedded systems. In our research, hardware may be varied and engulf systems such as mobile devices, robotics, or embedded integrated SoC (System-on-Chip). This search can yield valuable insights into what has worked or is missing regarding software solutions. To compare different embedded system solutions, we also want to identify metrics valued by the scientific community (i.e. power consumption, latency).

We plan to test existing solutions in available embedded devices. In particular, we initially consider two distinct hardware configurations: 1) Nvidia Jetson Nano board (embedded CPU+GPU); 2) Raspberry Pi. Regarding power, a consistent way of measuring energy consumption across different hardware systems is using a voltage and current tracker connected between the device and the power source. We have deployed the UM25C USB energy measurement device, which transmits its measurements via Bluetooth.

The final step is proposing or integrating existing solutions for handling parallelism, communication, work-load distribution, and others targeting embedded systems and the stream processing domain. From this point, we may identify patterns and lower-level APIs used by a higher-level coding abstraction. For that, we pretend to adapt a domain-specific language suitable for the stream processing domain called SPar[Griebler 2017]. With that, the programmer can rely on a lightweight programming model that uses semi-automatic code transformations to provide a higher-level interface for parallelism and service-level objectives. However, SPar has not been tested in the embedded systems domain, where high-level abstractions may benefit.

The research proposition of this work requires a further understanding of the requirements and limitations for stream processing applications in the context of embedded systems. Advancing this research may yield new insights for achieving high performance while easing the effort of handling parallelism in this environment. The results should increase productivity and portability at minimal performance costs.

References

Aldegheri, S. e. a. (2018). Enhancing performance of computer vision applications on low-power embedded systems through heterogeneous parallel programming. In *IFIP*.

Andrade, H. e. a. (2014). *Fundamentals of Stream Processing: Application Design, Systems, and Analytics*. Cambridge University Press, Cambridge.

Griebler, D. e. a. (2017). SPar: A DSL for High-Level and Productive Stream Parallelism. *Parallel Processing Letters*.

Lin, P. e. a. (2021). Resource management for pervasive edge computing-assisted wireless vr streaming in industrial internet of things. *IEEE Transactions on Ind. Inf.*