Processor-Specific Stream Processing Query Compilation

(Master project, Summer 2021)

Current Stream Processing Engines (SPEs) can process millions of records across hundreds of nodes to analyze an ever growing amount of real-time data. However, the most widely used SPEs, such as Apache Flink, Spark Streaming, or Storm are all JVM-based and do not utilize the servers' hardware efficiently. Recent work [1] shows that many hardware-specific optimizations can be made to improve the efficiency of each individual node (i.e., scale-up instead of scale-out). One such optimization is to compile queries instead of interpreting them. Grulich et al. [2] show that query compilation allows a single node to process orders of magnitude more records than unoptimized JVM-based systems.

In a general trend, non x86-based systems are rapidly catching up or have even surpassed Intel's CPU performance. Thus, investigating SPEs on these alternative systems may lead to surprising results. Additionally, each hardware systems allows for different optimizations based on its respective design choices. To fully utilize each system, each system needs to be understood and evaluated, before finally adopting system-specific implementations.

In this project, we will build on previous work (e.g. Grizzly or the current Masterproject) and investigate how query compilation in SPEs perform on a variery of systems. To this end, we will sytematically evaluate a SPE prototype on Intel, ARM, AMD, and PowerPC hardware to determine whether there are significant differences between these systems. In a second step, we will target system-specific optimizations for the variying hardware (e.g., specific SIMD instructions, memory reordering, data fetch/store instructions). Ideally, we will publish the results at a relevant conference in the field of data management.

Students will learn the inner workings of stream processors and data management systems in general, with a particular focus on query compilation. It is targeting students interested in acquiring skills in data management, stream processing, data flows, compilers, and low-level systems programming. The project will be implemented in C++ (possibly with some C in it).

General information and an introduction on stream processing can be found in the O'Reilly blog posts by Tyler Akidau [3, 4] and the stream processing book [5].

Grading

Courses applicable: ITSE (Masterprojekt), DE (Data Engineering Lab)

Graded activity:

- Implementation / group work
- Final report (8 pages, double-column, ACM-art 9pt conference format)
- Final presentation (20 min)

Contact

Lawrence Benson

Literature

[1]: Zeuch et al., 2019. Analyzing Efficient Stream Processing on Modern Hardware, *PVLDB*
[2]: Grulich et al., 2020. Grizzly: Efficient Stream Processing Through Adaptive Query Compilation,

SIGMOD
[3]: Tyler Akidau: Streaming 101. https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101
[4]: Tyler Akidau: Streaming 102. https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-102
[5]: Tyler Akidau, Slava Chernyak, Reuven Lax: Streaming Systems. O'Reilly. http://streamingsystems.net/