Research Statement

Caleb Stanford

March 2021

Overview

My research lies in applying programming languages and formal methods techniques to emerging data-driven computing applications.

I have particularly focused on the design and implementation of *distributed stream processing systems*, specialized software platforms for processing large quantities of data and responding in real time.

My research aims to answer three primary questions:

1. How can users best specify computations over streaming data?
2. What tools and techniques are available – and lacking – for ensuring such computations are correct?
3. What tools and techniques can be used to enable more efficient implementations of such computations?

Key Applications and Background

IoT . The data landscape of the near future will include a massive amount of data from Internet of Things devices, including health monitoring implants, drones, and regular smartphones. Many emerging applications rely on streaming analytics of such data in real time. A number of distributed data stream processing platforms are becoming popular for such analytics, including for example, Apache Flink, Apache Spark Streaming, Apache Storm, Twitter Heron, and IBM Streams. My research applies programming languages and formal methods techniques to improve the reliability, design, and implementation of such data stream processing platforms.

Runtime Monitoring . Large-scale software projects

Defined broadly, stream processing systems are specialized software platforms designed for processing large quantities of data and responding in real time. For instance, such software can be used to monitor financial transactions or stock prices, or to process input data from a distributed smart things environment, such as the network of devices in a smart home. Stream processing systems are popular because they allow the programmer to specify the computation in an intuitive way (e.g., as a high-level query, as a sequence of stream transformations, or as a dataflow graph), and the system will deploy and parallelize the computation automatically. Popular modern stream processing systems include Apache Spark Streaming and Apache Flink.

Reliability

To improve reliability, we have built tool support on top of Apache Storm (PLDI 2019) and Apache Flink (in submission) for testing and verification to prevent bugs due to data parallelism. Our work in Apache Storm shows that, by manipulating data streams with extra type information, programmers can statically ensure that their code parallelizes correctly. Our work in Apache Flink focuses on testing, showing that programmers can use similar annotations to automatically test applications for bugs due to parallelism, without having to modify the application source code.

Performance

On the design and implementation side, we have proposed techniques for compilation and optimization. For high-level query languages incorporating user-defined stateful and quantitative computations, we show that a new intermediate representation can be used to achieve efficient compilation with static bounds on performance (POPL 2019), and formally study the benefits of related program representations (ICALP 2017, TCS 2019). Our recent and ongoing work focuses on distributed compilation of stream processing applications in the internet of things domain: we have built a prototype stream processing system (in submission) for this problem which safely distributes the computation over many devices, while minimizing network load.

Other Research

Future Work

Test [1]

References

[1] Rajeev Alur, Konstantinos Mamouras, and Caleb Stanford. Modular quantitative monitoring. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–31, 2019.