Research Statement

Shuhao Zhang

July 1, 2020

In the investment community, the *time value of money* states that money is always more valuable today than in the future. A similar concept of *time value of data* is getting widely recognized in the data processing community – insights are derived from processing data, and the value of insights diminishes very fast with time. Due to the increased automation in many domains such as telecommunications, health care, transportation, and retail, numerous data-intensive applications are deployed in real-world use cases. Those applications generally involve continuously low-latency, complex analytics over massive data streams and are often named as *streaming applications*. Data stream processing system (DSPS) is a software that allows users to efficiently run their streaming applications in a scalable way.

Despite the successes achieved during the last several decades, DSPSs are now facing great challenges when supporting a wide range of emerging applications, which generally require the underlying DSPSs to achieve low end-to-end latency when processing huge volumes of data with complex computation and intensive state access. Witnessing the emergence of modern commodity machines with massively parallel processors, researchers and practitioners find shared-memory multicore architectures an attractive platform for DSPSs. However, fully exploiting the computation power delivered by multicore architectures are still challenging.

I am a system researcher with a computer science background in database design and software engineering. My research strives to build next generation of DSPSs and algorithms towards more efficient and scalable by exploiting the more and more widely accessible powerful hardware devices. For example, the system that I have built so far, for the first time, scales stream computation towards hundred of cores under the Non-Uniform-Memory-Access effect [5]. This brings the potential of accelerating existing stream processing application beyond what can be achieved today, and brings many new opportunities of supporting emerging data-intensive applications, such as online big-data machine learning.

In the coming years, I plan to further explore new optimization opportunities that improves the state-of-the-art approaches in managing big data streams including both more efficient systems and algorithms. This brings research interests involving not only system but also application research, and has the potential to make large impact in both academic and industry community. In the following, I will summary my past key research activities, which are first-authored published in top-tier conference in the database community. Beyond that, I have also first-authored two patents [4, 7] registered in US based on my past research results, which indicates the large potential of industry and society impact of my research. I will end this article with my future research plans.

Multi-Query Optimization for Complex Event Processing in SAP ESP (ICDE'17)

As a PhD scholar in SAP Innovation Center Singapore from 2014 to 2018, I participated in improving SAP's stream processing platform, called SAP ESP. The system aims at delivering real-time stream processing and analytics in time-critical applications. In SAP ESP, users can implement their complex event processing tasks, which continuously analysis real-time event streams and quickly identify pre-defined complex events. I have created MOTTO [6, 7], a multi-query optimizer for complex event processing in SAP ESP as illustrated in Figure 1. MOTTO realizes more sharing opportunities by introducing pattern query decomposition and transformation. Those sharing techniques are also extented to support multiple nested pattern queries and pattern queries with different window constraints. Experiments demonstrate the efficiency of MOTTO with both real-world applications scenarios and sensitivity studies.

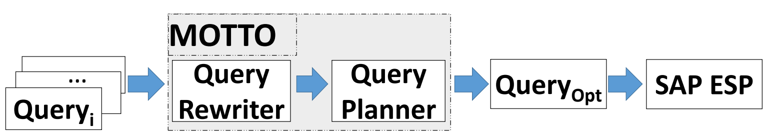
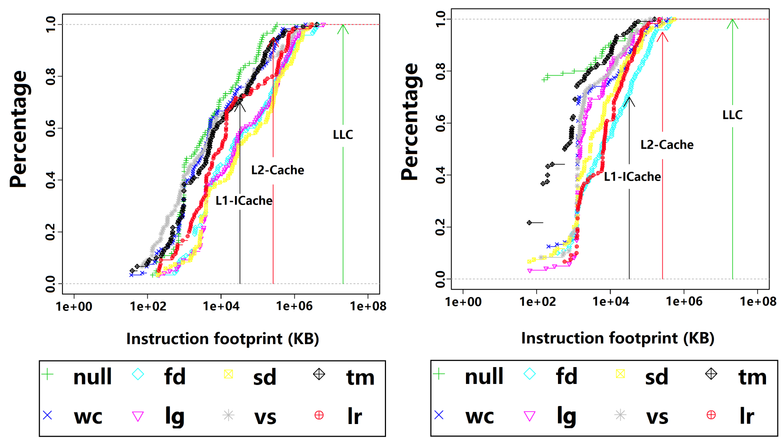


Figure 1: Multi-query optimization workflow of MOTTO.

Rvisiting the Design of Data Stream Processing Systems on Multi-Core Processors (ICDE'17)



(a) Storm (b) Flink

Figure 2: Instruction footprint between two consecutive invocations of the same function.

For my Ph.D. dissertation, I was pioneering in discover the gaps between the design of modern stream processing systems and modern hardware architectures. In particular, I summarize [3] three common design aspects of modern DSPSs, including a) pipelined processing with message passing, b) on-demand data parallelism, and c) JVM-based implementation. Then, I conducted detailed profiling studies with micro benchmark on modern multi-socket multi-core by using Apache Strom and Flink as examples. The results have shown that those designs have underutilized the scale-up architectures in these two key aspects: a) The design of supporting both pipelined and data parallel processing results in a very complex massively parallel execution model in DSP systems, which causes high front-end stalls on a single CPU socket; b) The design of continuous message passing mechanisms between operators severely limits the scalability of DSP systems on multi-socket multi-core architectures. For a concrete example, Figure 2 illustrates that the instruction footprint of both Storm and Flink exceed L1-Instruction cache, and hence leads frequent cache trashing. Based on the profiling results, I have further proposed two optimizations [4] and demonstrate promising performance improvements.

BriskStream: Scaling Data Stream Processing on Shared-Memory Multicore Architectures (SIGMOD'19)

My previous profiling study shows that existing DSPSs underutilized the underlying complex hardware microarchitecture and especially show poor scalability due to the unmanaged resource competition and unaware of NUMA effect. Hence, my subsequent effort spend on a complete revolution in designing next-generation stream processing platform, namely BriskStream [5], specifically optimized for sharedmemory multicore architectures. To address NUMA effect, I have developed a new streaming execution plan optimization paradigm, namely Relative-Location Aware Scheduling (RLAS). Figure 3 shows the better scalability of BriskStream than existing popular DSPSs on multi-socket servers by taking Linear-Road Benchmark as an example. Unmanaged thread interference and unnecessary remote memory access penalty prevent existing DSPSs from scaling well on the modern multisockets machine. The comprehensive experiments based on two eight-sockets machines confirm that BriskStream significantly outperforms existing DSPSs up to an order of magnitude even without the tedious tuning process. In short, I showed how a DSPS, for the first time, scales stream computation towards a hundred of cores under NUMA effect.

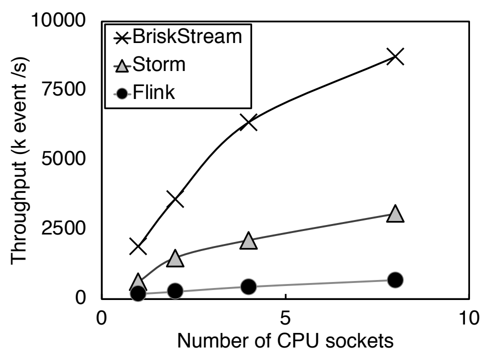


Figure 3: System scalability comparison based on Linear-Road Benchmark.

Towards Concurrent Stateful Stream Processing on Multicore Processors (ICDE'20)

DSPS with transactional state management relieves users from managing state consistency by themselves, and has recently received attention from both academia and industry community. However, scaling stream processing while providing transactional state management on modern multicore processors is challenging. On the one hand, to achieve both low latency and high throughput, DSPSs can process multiple input events at the same time in order to aggressively exploit parallelism. On the other hand, processing different events concurrently may lead to conflict accesses (reads and writes) to the same application state, hence leading to higher chances of violating transactional state consistency. To make things worse, more than simply guaranteeing the ACID properties preserved in the relational database systems, DSPSs further need to enforce the state access *order* according to the input event sequence. Witnessing those issues, I have developed TStream [8], a new DSPS that can support highly scalable stream processing with transactional state consistency guarantee on multicores. In order to take advantage of multicore architectures, TStream detaches the state management from the streaming computation logic, and performs its internal state maintenance asynchronously. By eliminating the expensive synchronization primitives, TStream aggressively extracts parallelism opportunities by revealing the operation dependencies at runtime. The initial results show that TStream achieves several times higher throughput on average over existing solutions with similar or even smaller end-to-end processing latency.

Research Agenda

The trend of more radical performance demand, complex analytics, and intensive state access have accelerated the development of next-generation data stream processing systems (DSPSs) [3, 5, 8]. Beyond what I have previously described, there are a number of emerging issues in data stream processing that we shall pay more attention to.

The Internet of Things (IoT) presents a novel computing architecture for data management. In such context, the ability to scale both out and up is crucial to effectively improve performance by orders of magnitude. How to use the more and more heterogeneous devices in the present of millions distributed entities remains unclear today. My short term plan (2∼3 years) is hence to collaborate with my current postdoc advisor, Prof. Markl Volker (TU Berlin) to investigate the future stream processing system [9] designed for the IoT environment. Specifically, I plan to explore questions such as:

1. **Privacy and security.** In an IoT environment, data processing at the edge is highly exposed to security threats, e.g., a sensor may be hacked. Intel SGX is one of the popular Trusted Execution Environment (TEE) implementations, which may help to provide security guarantees to edge processing. The first step is to investigate the designing of important stream operations (e.g., filter, projection, windowing) running based on Intel SGX.

The potential long term goal of this direction is a redesign towards more secure data processing platforms in the context of IoT environment [2].

2. **Transactional state management.** Recent DSPSs can achieve excellent performance when processing large volumes of data under tight latency constraints. However, they sacrifice support for concurrent state access that eases the burden of developing stateful stream applications. One of the promising ways of managing concurrent state access during stream processing is to model state accesses as transactions. My current result [8] successfully achieves orders of magnitude improvement over the state of the art on shared-memory multicore processors. However, many questions are still remain unsolved, such as how to achieve the same performance improvement in the context of distributed environment. Another potential weakness is its persistency from crash failures. In contrast to traditional database, stream processing often needs to preserve event process ordering. How to efficiently provide high concurrency and fast durability to DSPSs while preserve event process ordering remains a non-trivial question.
   The potential long term goal of this direction is a redesign towards a more reliable and deterministic transactional data stream processing system.
3. **Machine learning and Data Mining.** The principal task of online machine learning is to learn a concept incrementally by processing labeled training examples one at a time. Although the massively parallel processors of modern hardware provides the opportunities in providing very fast training process, there is little work on exploring how to accelerate future DSPSs in supporting real-time machine learning activities. Fast data exploration has also received many attentions recently [1]. It aims to present users preliminary results as fast as possible in order to improve user interactivity. However, little works or systems are able to achieve pleasant results, there are still huge potential to explore. For example, how to answer applications with user-defined functions quick remains an open question.
   The potential long term goal of this direction is a redesign towards a more functional and convenient data stream processing system that supports online machine learning and fast data exploration.

References

[1] J. Ang, T. Fu, J. Paul, **S. Zhang**, B. He, T. S. D. Wenceslao, and S. Y. Tan. Trav: An interactive exploration system for massive trajectory data. In *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, pages 309–313, Sep. 2019.

[2] H. Park, S. Zhai, L. Lu, and F. X. Lin. Streambox-tz: Secure stream analytics at the edge with trustzone. In *Proceedings of the 2019 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '19, page 537–554, USA, 2019. USENIX Association.

[3] **Shuhao Zhang**, B. He, D. Dahlmeier, A. C. Zhou, and T. Heinze. Revisiting the design of data stream processing systems on multi-core processors. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*, pages 659–670. IEEE, 2017.

[4] **Shuhao Zhang**, B. He, and D. H. R. Dahlmeier. Efficient execution of data stream processing systems on multi-core processors, May 10 2018. US Patent App. 15/348,932.

[5] **Shuhao Zhang**, J. He, A. C. Zhou, and B. He. Briskstream: Scaling Data Stream Processing on Multicore Architectures. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, Amsterdam, Netherlands, 2019. ACM.

[6] **Shuhao Zhang**, H. T. Vo, D. Dahlmeier, and B. He. Multi-query optimization for complex event processing in sap esp. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1213–1224. IEEE, 2017.

[7] **Shuhao Zhang**, H. T. Vo, D. H. R. Dahlmeier, and B. He. Multi-query optimizer for complex event processing, Apr. 24 2018. US Patent 9,953,056.

[8] **Shuhao Zhang**, Y. Wu, F. Zhang, and B. He. Towards concurrent stateful stream processing on multicore processors. ICDE '20.

[9] S. Zeuch, A. Chaudhary, B. Del Monte, H. Gavriilidis, D. Giouroukis, P. M. Grulich, S. Bress, J. Traub, and V. Markl. The nebulastream platform: Data and application management for the internet of things. CIDR '20.