Correctness in Stream Processing
Challenges and Opportunities

Caleb Stanford, Konstantinos Kallas, and Rajeev Alur
Stream Processing

Streaming Data → Real-Time Decisions → Applications

Companies:
- databricks
- aws KINESIS
- Materialize
- Spark
- Apache Flink
- Apache Storm

Systems:
Stream Processing

**Formal correctness support?**

“The nature of debugging is therefore post-mortem. Developers are notified of runtime failures or incorrect outputs after many hours of wasted computing cycles on the cloud.”

- [Gulzar et. al, Bigdebug, 2016]
- [Vianna et. al, testing in data stream processing applications, 2019]
Challenges

**No unified language standard**

- Dataflow graph edges: ordered or unordered?
- Stream partitioning: annotated or inferred?
- Complex features:
  - stateful operators, external services, iterative computation

(Contrast with: traditional relational algebra)

**Unified semantics is a precursor to all verification tools**
Opportunities

Correctness dimensions common to all systems

1. Order-aware computation
2. Correct distribution (beyond sharding)
3. Performance guarantees
4. Fault tolerance
Vision

User Application

Stream Processing System

Formal Execution Semantics: Annotated Dataflow

Compiler/Optimizer

Distributed Implementation

Distributed Assumptions Analysis

Ordering Requirements Analysis

Performance Analysis

Formal Analyses