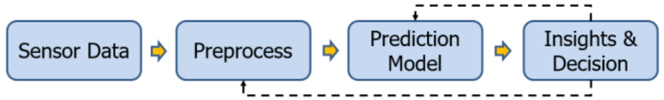
StreamQL: A Query Language for Efficient Data Stream Processing

Lingkun Kong, Konstantinos Mamouras

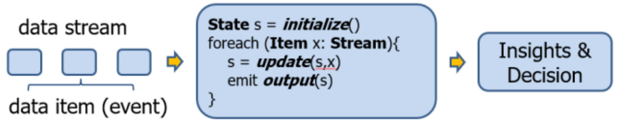
Rice University

Motivation

- ***Big-data era:*** IoT applications such as *predictive maintenance* collect, process and analyze a massive amount of data in real-time.



- The setting of **data stream processing**
  - unbounded source of data
  - real-time, very high rate
  - complex patterns
- Low-level stream processing using general-purpose programming language is **cumbersome, error-prone**, and **not modular.**



State of the Art

Streaming Database: STREAM, Aurora, Borealis, CACQ, TelegraphCQ, Niagara, Gigascope, Nile, StreamInsight

Complex Event Processing: SQL-TS, SASE, Cayuga, SPL MatchRegex, Zstream, Trill, Esper, Siddhi, Flink, Oracle Stream Analytics, IBM Streams

Distributed Stream Processing: S4, IBM Streams, MapReduce Online, Storm, Heron, Samza, Naiad, Spark Streaming, Flink, Google's MillWheel, Apex

Synchronous Dataflow: SIGNAL, Esterel, LUSTRE, StreamIt

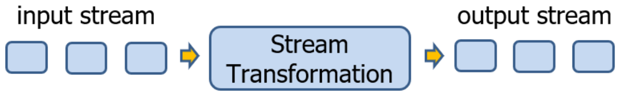
Real-time Signal Processing: Xstream, WaveScope, TrillDSP

Light-weight Streaming Engine: Microsoft's Trill, Esper, Siddhi, ReactiveX, StreamQRE

- ***Limitations***:
  - Lack of stream abstractions and formal semantics.
  - No guarantee of correctness.
  - Inefficient in detecting complex patterns.

Our Approach

- **StreamQL** (Streaming Query Language) simplifies the task of specifying complex streaming computations.
  - **Stream processing is a procedure that transforms the input stream to the output stream.**



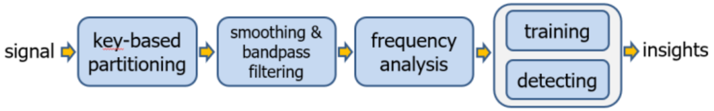
- ***Contributions***:
  - **economical** (only half size of RxJava)
  - proved to be **expressive** and **correct.**
  - **better throughput performance** in practice in comparison to other state-of-the-art approaches.
  - **signal processing** and **machine learning** toolbox.

Relational	Dataflow	Temporal
map, filter, emit, aggr, groupBy, window	compose, parallel	take, skip, search, seq, iterate

Case Studies

- ***Predictive maintenance:***
  - Rolling Bearing Fault Prediction
  - Battery Aging
- ***Healthcare Monitoring:***
  - Cardiac Signal analysis
  - Arterial Blood Pressure monitoring
  - Walking motion detection
- ***High-frequency Market Analysis:***
  - Trading direction analysis

- ***Example of Predictive Maintenance:***
  - pipeline-styled streaming computation
  - windowing and key-based partitioning
  - signal processing support
  - machine learning support




```
getMagn = map(x -> sqrt(x.ax*x.ax+x.ay*x.ay+x.az*x.az)/3.0);
smoothing = FIR([... fir parameters ...]);
bandpass = IIR([... bandpass filtering parameters ...]);
envelope = HT(... Hilbert transform parameters ...) >> abs();
spectrum = window(n, s, FFT(... FFT parameters ...));
getFeatures = getMagn >> smoothing >> bandpass >>
              envelope >> spectrum;
training = SVM(... training parameters ...);   detecting = ... ;
singleProc = getFeatures >> seq(training, detecting);
process = groupBy(x -> x.id, singleProc, (key,res) -> ... );
```

Experiments

- The Java implementation of StreamQL is evaluated with RxJava and Siddhi in one micro benchmark and four benchmarks with realistic workloads.
  - ***Micro Benchmark***: For basic stream computations, StreamQL is **1.1-100 times faster than RxJava** and **2–100 times faster than Siddhi**.
  - ***Realistic Workloads***: For computations involving complex streaming aggregation and pattern detection, StreamQL is on average **5 times faster than RxJava** and **40 times faster than Siddhi**.