# Jumping the ORDER BY barrier in large-scale pattern matching

## ABSTRACT

## 1. INTODUCTION

(Motivation) Event-series pattern matching is widely used for behavioral/anomaly/causality analysis, for eg: network diagnostics, algorithmic trading, security breach detection.

(Challenges) Warehoused data is being analysed by a wide range of queries thus it may not be clustered/sorted on time.

Analytics are being run both over fresh data as well as historical data, therefore the data management system needs to support both online and batching mode processing.

(Approach) We design a set of relational-style optimizations for event-series pattern matching workloads.

(Results) We provide X reduction in shuffled data and Y reduction in end-to-end latency for 3 workloads: i) telemetry analysis over the events produced by the Windows event-reporting infrastructure ii) Github queries iii) stock market algorithmic trading.

## 2. RELATED WORK

(*) Pattern matching over out-of-order event series.
(*) Distributed pattern matching
(*) Stream processing engines optimizations
(*) Pattern matching algorithms optimizations

## 3. DESIGN

We consider patterns specified as a symbolic finite automata where each transition has a guard (propositional formula) whose predicates test the currently considered event against constants as well as events matched by preceding transitions.

Pattern matching algorithms require data sorted on time. In a map-reduce framework this makes the reducer the main bottleneck of the analysis, both at the network level (large amaounts of shuffled data) and at the processing level (in case of data-skew).

We propose to significantly reduce the amount of data fed into the pattern matcher (reducer) by dropping in a map phase all the events that cannot take part in a successfull match.

To do so we design a set of filters that while conservative, closely match the semantics of the patterns analysed. We propose three levels of abstraction. The first enforces the join constraints between different transitions as expressed by join predicates within the transition guards. The second one further imposes time windowing constraints (all events of a successfull match must occur within a timeout of the first event in the match). Finally the last one enforces ordering constraints between *consecutive* transitions of the pattern.

In our approach we leverage the fact that a large class of symbolic finite automata fit within the fragment of star-free languages which have a corresponding FOL formula (possibly with counting quantifiers). Whenever that is not the case we can narrow the scope of our filters to the sections of the pattern that do.

We relax these FOL formulas to produce precise filters, one per transition, that can be evaluated independently on each mapper. Finally we further coarsen these filters to a set of relaxed filters that can be collected and tested in a time and space efficient manner (eg. bloom filters, time-interval maps).

We explore the tradeoffs involved in the overheads incurred in building the filters and their accuracy.

## 4. IMPLEMENTATION

## 5. EVALUATION

We tested our approach on 3 workloads: i) Asimov, consisting of telemetry events and Asimov patterns (3700 individual patterns + all the patterns combined in a single query) ii) SOSP queries over Github dataset iii) Cayuga queries over stock trades dataset.

## 6. CONCLUSIONS

## 7. REFERENCES



© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235