Towards Creating a Generalized Complex Event Processing Operator Using FlinkCEP: Architecture & Benchmark

Eleni Kougioumtzi
elkoum15@gmail.com
Athens University of Economics and Business

Antonios Kontaxakis
Antonios Deligiannakis
{adokondax,adel}@athenarc.gr
Athena RC & Technical Univ. of Crete

Yannis Kotidis
kotidis@aueb.gr
Athens University of Economics and Business

ABSTRACT
FlinkCEP, the Complex Event Processing (CEP) API of the Flink Big Data platform, scales-out pattern detection to a number of machines in a computer cluster or cloud. The high expressive power of FlinkCEP’s language comes at the cost of cumbersome parameterization of the patterns to be monitored, thus limiting usability. In this work, we build a novel, logical CEP operator that receives as input specifications of CEP queries in the form of extended regular expressions and automatically re-writes them to FlinkCEP programs. We also initiate a benchmarking effort on FlinkCEP.

CCS CONCEPTS
• Computer systems organization → Real-time systems.

KEYWORDS
Complex Event Processing, Flink

ACM Reference Format:

1 INTRODUCTION
Complex event processing (CEP) elevates traditional stream processing by allowing analysts to define and query for patterns in one or multiple streams [2]. These patterns constitute Complex Events (CEs) which consolidate incoming low-level, simple event observations based on their content, frequency and ordering relationships. CEs represent meaningful business rules, triggers or alerts that are useful in various business domains. For instance, CEP is often used to identify frauds in the financial or telecom domain [4] or threats at sea in the maritime domain [6], by aggregating vast and potentially conflicting streams of simple events.

CEs consist of simpler event patterns in the form of singleton, looping, sequencing or other patterns with the use of quantifiers. Queries for detecting patterns which represent the CEs additionally include (a) contiguity conditions (often called selection strategies [2]), (b) event consumption policies, for instance to denote how many matches an event may be assigned to, and (c) windowing operations [2]. More generic CEP patterns monitor correlations among CEs [5] that are even harder to express and debug programmatically. Therefore, modern CEP engines should provide a powerful CEP language with high expressiveness to allow the definition of complex query patterns along with their contiguity, consumption, and windowing conditions.

Recently, the developers of Flink have published FlinkCEP [1], a CEP library implemented on top of Flink. FlinkCEP allows the user to monitor and query for existing CEs over streams of events and the execution of these queries leverages the virtues of parallel processing capabilities of Flink to scale-out any CE detection effort over powerful computer clusters. The FlinkCEP library constitutes a great leap in deploying complex CEP workflows using a Flink cluster. However, at its core it is a programming library that requires user expertise in FlinkCEP.

In the INFORE project (Best Demo Award in CIKM2020) [3] we have been working on supporting non-expert programmers in performing optimized, cross-platform, streaming analytics at scale. Via the use of graphical tools (RapidMiner Studio), users can define streaming data workflows. Then, the INFORE optimizer instructs the deployment and execution of these workflows across Big Data clusters. As a new part of the work in [3], we here present a novel, generic INFORE CEP operator we have developed that takes as input the description of a complex pattern in the form of commonly used regular expressions, the desirable (contiguity, consumption, windowing) specifications and seamlessly transforms the specifications of this INFORE CEP pattern to a FlinkCEP program that can be automatically submitted for execution to any Flink cluster. This enables users with no programming experience to rapidly define their business rules and directly deploy them for monitoring. We further provide experiments showcasing the performance of the
were instantiated as Kafka topics using a broker running in the 18.04 using the latest version of Flink. The input and output streams and injected 125K CEs at random positions within the input streams.


Each stream was processed using tumbling windows of 128 events each and the default consumption policy was NO_SKIP.

In Figure 2 we depict the throughput (processed input events/sec) as we vary the degree of parallelism of the INFORE CEP operator and the desired contiguity condition. We notice that all instances of the operator scale by increasing the parallelism except for the execution of the strict contiguity with parallelism = 8, as in that case, the speed of the operator is capped by the rate of reading data from Kafka. As expected, selecting the relaxed or the non-deterministic contiguity conditions results is significantly smaller throughput due to increased number of partial pattern matches monitored before detecting CEs and the larger number of CEs produced.

In Figure 3 we repeat the experiment with parallelism = 1 and vary the contiguity condition and consumption policy. Using a consumption policy that skips events after a match benefits mainly the relaxed and non-deterministic conditions, as it helps limit the number of partial matches monitored and CEs produced.

4. CONCLUSIONS

We presented the INFORE CEP operator that accepts event pattern queries in the form of commonly used regular expressions along with event contiguity, consumption and windowing specifications. INFORE CEP seamlessly interprets these patterns to FlinkCEP programs, enabling rapid submission of CEP jobs to clusters or clouds of choice. Our future work is on extending the preliminary benchmarks of this work towards a learning-based cost estimator.

ACKNOWLEDGMENTS

This work has received funding from the EU Horizon 2020 research and innovation program INFORE under grant agreement No 825070.

REFERENCES


