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,adeli}@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.

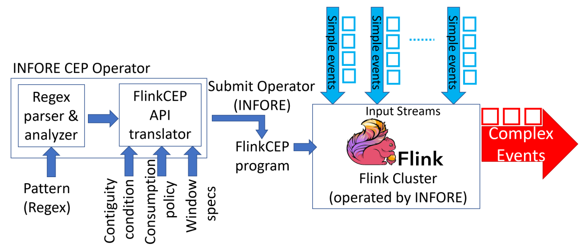

Figure 1: INFORE CEP Operator to FlinkCEP Program.

CCS CONCEPTS

• **Computer systems organization → Real-time systems**.

KEYWORDS

Complex Event Processing, Flink

ACM Reference Format:
Eleni Kougioumtzi, Antonios Kontaxakis, Antonios Deligiannakis, and Yannis Kotidis. 2021. Towards Creating a Generalized Complex Event Processing Operator Using FlinkCEP: Architecture & Benchmark. In *The 15th ACM International Conference on Distributed and Event-based Systems (DEBS '21), June 28-July 2, 2021, Virtual Event, Italy*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3465480.3467841

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

DEBS '21, June 28-July 2, 2021, Virtual Event, Italy

ACM ISBN 978-1-4503-8555-8/21/06…$15.00
https://doi.org/10.1145/3465480.3467841

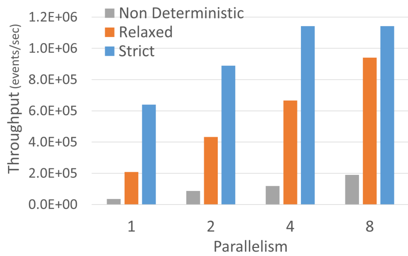

Figure 2: Varying Parallelism & Contiguity.

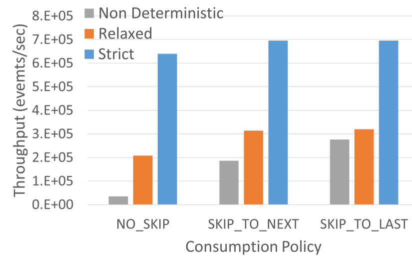

Figure 3: Varying Contiguity & Consumption.

transformed FlinkCEP programs under different pattern (contiguity, consumption, windowing) specifications.

2 DESCRIPTION OF INFORE CEP OPERATOR

Figure 1 depicts the process of creating an INFORE CEP operator up to the point of submitting a FlinkCEP job to a cluster. The user provides (i) a regular expression (Regex) denoting the complex event to be recognized, (ii) the contiguity condition, (iii) the consumption policy and (iv) the window specification. The operator is submitted by the INFORE optimizer to the cluster. Each operator can monitor one or multiple input streams, while its degree of parallelism can also be adjusted.

The parser processes a Regex that may include disjunction, negation, quantifiers and the wild char. For example $a\{2,5\}b?(c|d)$ denotes that we seek between 2 and 5 appearances of event a followed by zero or one event b followed by events c or d. The Regex is translated into a FlinkCEP program in accordance with the selected contiguity and consumption policies, discussed next.

FlinkCEP supports the following contiguity conditions (i) Strict Contiguity: where matching events appear strictly one after the other, (ii) Relaxed Contiguity: where non-matching events appearing in-between the matching ones are ignored, and (iii) Non-Deterministic Relaxed Contiguity: that allows non-deterministic actions between matching events. For event consumption, FlinkCEP provides the options: (i) NO_SKIP: produce all matches (ii) SKIP_TO_NEXT: discard partial matches that started with the same CE event (iii) SKIP_PAST_LAST_EVENT: discard partial matches after CE match started but before it ends. Two more options are SKIP_TO_FIRST[p] and SKIP_TO_LAST[p] that are similar to SKIP_TO_NEXT and SKIP_PAST_LAST_EVENT, respectively, but they use a pattern p to dictate the start (resp. last) event in the CE [1].

3 EXPERIMENTS

In this section we provide experimental results of the prototype INFORE CEP operator we developed.[1] All experiments were run in a VM with 16 cores and 16GB of main memory running Ubuntu 18.04 using the latest version of Flink. The input and output streams were instantiated as Kafka topics using a broker running in the same machine. We used 8 streams with 2M events each (16M in total). Each stream was populated using randomly selected events denoted as lower case Latin characters. We used Regex $ab\{1,3\}c|d$ 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 window 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

[1] FlinkCEP. [n.d.]. https://ci.apache.org/projects/flink/flink-docs-release-1.12/dev/libs/cep.html. [Online; accessed 13-April-2021].

[2] N. Giatrakos, E. Alevizos, A. Artikis, A. Deligiannakis, and M. N. Garofalakis. 2020. Complex event recognition in the Big Data era: a survey. *VLDB J.* 29, 1 (2020), 313–352. https://doi.org/10.1007/s00778-019-00557-w

[3] N. Giatrakos, D. Arnu, T. Bitsakis, A. Deligiannakis, M. N. Garofalakis, R. Klinkenberg, A. Konidaris, A. Kontaxakis, Y. Kotidis, V. Samoladas, A. Simitsis, G. Stamatakis, F. Temme, M. Torok, E. Yaqub, A. Montagud, M. P. de Leon, H. Arndt, and S. Burkard. 2020. INforE: Interactive Cross-platform Analytics for Everyone. In *CIKM*.

[4] N. Giatrakos, A. Artikis, A. Deligiannakis, and M. N. Garofalakis. 2019. Uncertainty-Aware Event Analytics over Distributed Settings. In *DEBS*.

[5] A. Grez and C. Riveros. 2020. Towards Streaming Evaluation of Queries with Correlation in Complex Event Processing. In *ICDT*.

[6] A. Troupiotis-Kapeliaris, K. Chatzikokolakis, D. Zissis, and Elias Alevizos. 2020. Experimental Comparison of Complex Event Processing Systems in the Maritime Domain. In *MDM*. 293–298.

[1] Code available at: https://github.com/eleniKougiou/Flink-cep-automation