Streaming Process Mining with Beamline

Andrea Burattin
DTU Compute – Technical University of Denmark
andbur@dtu.dk

***Abstract*—Beamline is a Java framework designed to facilitate the prototyping and development of streaming process mining algorithms. The framework is designed on top of Apache Flink which makes it suitable for extremely efficient computation due to its distributed and stateful nature. Beamline consists of both algorithms as well as data structures, sources, and sinks to facilitate the development of process mining applications. The framework is licensed with Apache-2.0 and its companion website https://www.beamline.cloud contains real-life examples on actual live data and all the system's documentation.**

***Index Terms*—Streaming Process Mining, Apache Flink, Event stream**

I. Introduction

Process mining [1], [2] is a family of techniques aiming at constructing abstract models (e.g., Petri nets [3], [4]) and verifying process executions with the final aim of understanding how these processes are performed, starting from event logs (i.e., recording of what happened).

Process mining is typically divided into several sub-tasks including *control-flow discovery* [1] aiming at discovering a control-flow model starting from executions of the model itself; *conformance checking* [5], aiming to verify that the executions of a process are conforming a normative process description. Real-world application examples of control-flow discovery could aim at understanding how a firm manufactures or handles goods (with the goal of understanding the *in-vivo* processes, to optimize them); applications of conformance checking could target clinical protocols and ensure that these are aligned with the expected protocols (with the goal of spotting patients' mistreatments as soon as possible).

Process mining has been applied in many disciplines and, one of the most impactful applications, right now, is in the healthcare [6] where clinical protocols/guidelines are the *processes* and treatments of patients are the executions, or *event logs*. Particularly in this domain, a fundamental requirement is the ability to change the course of treatment while the patient is being medicated, thus requiring a streaming (or online) analysis (as opposed to a historical, or offline, analysis).

Streaming data analysis [7] comes with a set of computational requirements that are directly transferred into the streaming process mining discipline [8]. In addition to these, in the latter, the fact that many data points – each of them observed at different timestamps – should be conceptually connected to each other introduces some complexity based on the observation window (i.e., the period of time during which the analysis is performed) [9].

The software presented in this paper, called Beamline, which is built on top of Apache Flink[1] [10], enables the implementation of streaming data and process mining pipelines, by providing access to the streaming process mining algorithms as well as common data analysis techniques.

II. Overview and Design

Beamline is defined as an extension of Apache Flink. The latter is a library for distributed stateful computations over data streams. Specifically, Apache Flink allows the definition of pipelines called *dataflow* that define which manipulations each event is expected to go through. Beamline is a set of operations that extends the capabilities of Apache Flink, including process mining transformations, such as process-aware event filters or flat-mappers for the discovery of processes or the computation of the conformance.

Due to the fact that Beamline is an extension of Apache Flink, all event transformations (both pre- and post-processing) and all the data connectors implemented are accessible.

III. Functionalities available

While Beamline is designed as a tool for researchers and practitioners for developing and deploying new streaming process mining algorithms, a lot of functionalities are available off-the-shelf, thus resulting in the ability to immediately benefit from the tool.

It is possible to ingest events using all Apache Flink connectors. In addition, for testing purposes, it is also possible to "replay" static logs as well as to simulate events referring to known processes using the PLG2 simulator [11]. Once events are imported into the platform, some process-aware filters are available, for example, to filter (retain/exclude) events based on specific activities, process instances, or other event properties.

The first option to consume an event stream consists of performing control flow discovery, i.e., producing a process representation that captures a process expressing all events currently being observed. It is important to note that this representation can evolve over time. On top of this representation different dimensions could be added as well, for example, the average time required to execute an activity or the maximum time between two activities, thus enabling to identify and locate bottlenecks. For example, imagine the production process employed in a frozen food factory. It is reasonable to think that such a process will be periodically

[1]https://flink.apache.org/

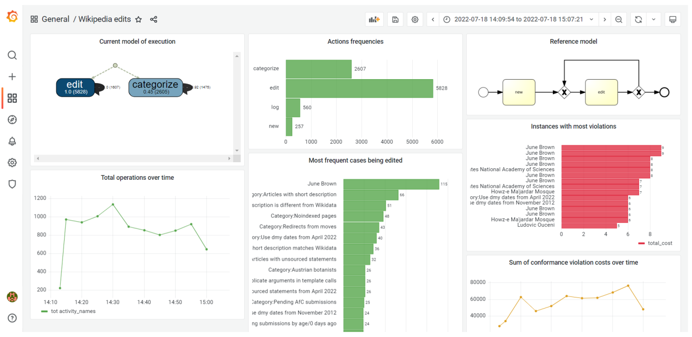
General / Wikipedia edits
Current model of execution
edit
categorize
Actions frequencies
Reference model
Instances with most violations
Total operations over time
Most frequent cases being edited
Sum of conformance violation costs over time

Fig. 1. A screenshot of Grafana showing data computed with Beamline.

switching between icecreams (during the months approaching summer) and frozen pizza (during the rest of the year). In this case, the changes will not involve only the control-flow but the frequencies as well. Beamline supports the discovery of processes using different algorithms, producing both imperative (e.g., using the Heuristics Miner with Lossy Counting) and declarative (e.g., with the Declare Discovery) models.

Another way of consuming an event stream is to perform conformance checking. This means providing a normative (i.e., a prescriptive) model and checking, for each event, whether the process instance being executed is conforming or not to the requirement. Meaningful use cases for this activity are, for example, in healthcare, where clinical guidelines should be followed but, as soon as violations are detected, alerts can be provided, to require a second look at the case and verify that the patient is treated properly. Beamline supports conformance checking where normative models are specified using the Petri net notation.

It is important to highlight that all results produced by Beamline can be sink-ed into any other system. For example, it is possible to forward the results of the computation into a time-series database (such as InfluxDB) for visualization with "observability platforms" (such as Grafana) as shown in Fig. 1. The website of Beamline as well as the GitHub repository provides examples of all the operations mentioned in this section (including the storage of results in an external database).

IV. Installation and Usage

The Beamline framework is hosted on GitHub[2], with its interactive documentation hosted on GitHub Pages[3], and installation instructions as well as many tutorials and "hands-on" real examples available on the project website[4]. It is possible to use Beamline on any Java project where dependencies are managed using either Gradle, Maven, `sbt`, or Leiningen. Beamline comes with all modules and extensions already compiled, therefore it is enough to just include the proper dependency and all necessary packages are automatically included.

[2]https://github.com/beamline/framework/

[3]https://beamline.github.io/framework/

[4]https://www.beamline.cloud/

V. Comparison to Related Software

While several other open-source software for process mining are available, such as ProM[5] [12] or PM4Py[6] [13], however their capability of handling streaming data is not (or only very partially) developed. Previous implementations of streaming process mining algorithms have been carried on using *ad hoc* software, hence making comparisons across techniques and algorithms extremely complicated.

When considering streaming data mining and streaming machine learning, several systems have been developed in the past, such as MOA [7] or Apache Flink [10]. While leveraging these is extremely important, as they already benefit from a huge community, none of them implement any process mining capability.

VI. Conclusion

Beamline is a Java framework designed to facilitate the prototyping and development of streaming process mining algorithms. Thanks to its integration into Apache Flink, users can leverage all capabilities of the latter platform to handle pre- and post-processing needed for their streaming (process) mining challenges.

A link to a screencast is available at https://youtu.be/8eagbpJ_hK4.

References

[1] W. M. van der Aalst, *Process Mining*. Springer, 2016.

[2] IEEE Task Force on Process Mining, "Process Mining Manifesto," in *Business Process Management Workshops*, F. Daniel, K. Barkaoui, and S. Dustdar, Eds. Springer-Verlag, 2011, pp. 169–194.

[3] W. M. van der Aalst, "Putting high-level Petri nets to work in industry," *Computers in Industry*, vol. 25, no. 1, pp. 45–54, 1994.

[4] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[5] J. Carmona, B. van Dongen, A. Solti, and M. Weidlich, *Conformance Checking*. Springer International Publishing, 2018.

[6] J. Munoz-Gama et al., "Process mining for healthcare: Characteristics and challenges," *Journal of Biomedical Informatics*, vol. 127, 3 2022.

[7] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive Online Analysis Learning Examples," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[8] A. Burattin, "Streaming Process Discovery and Conformance Checking," in *Encyclopedia of Big Data Technologies*, S. Sakr and A. Y. Zomaya, Eds. Springer International Publishing, 2018, pp. 1–8.

[9] ——, "Streaming Process Mining," in *Process Mining Handbook*, W. M. van der Aalst and J. Carmona, Eds. Springer, 2022, pp. 349–372.

[10] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache Flink™: Stream and Batch Processing in a Single Engine," in *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 2015, pp. 28–38.

[11] A. Burattin, "PLG2 : Multiperspective Process Randomization with Online and Offline Simulations," in *Online Proceedings of the BPM Demo Track 2016*. CEUR-WS.org, 2016.

[12] E. H. M. W. Verbeek, J. Buijs, B. van Dongen, and W. M. van der Aalst, "ProM 6: The Process Mining Toolkit," in *BPM 2010 Demo*, 2010, pp. 34–39.

[13] A. Berti, S. J. van Zelst, and W. M. van der Aalst, "Process Mining for Python (PM4Py): Bridging the Gap between Process-and Data Science," in *Proc. of ICPM Demo Track*, 2019.

[5]https://www.promtools.org/

[6]https://pm4py.fit.fraunhofer.de/