Rust4PM: A Versatile Process Mining Library for When Performance Matters

Aaron Küsters[1], Wil M.P. van der Aalst[1]

[1]Chair of Process and Data Science (PADS), RWTH Aachen University, Germany

Abstract

Rust4PM provides an open-source software library for process mining focused on performance. For instance, it supports parsing the most common data formats for standard and object-centric event logs (XES and OCEL 2.0) significantly faster than other available process mining tools. The library is written in the compiled, memory safe, programming language Rust, which focuses on performance and explicit error handling. As such, it is a good match for processing huge event data and other computationally expensive tasks or algorithms. Rust4PM aims to form a solid basis for new process mining software that emphasize execution speed or reliability, written in either Rust or another language (e.g., Python, Java, or JavaScript) via the use of appropriate language bindings.

Keywords

Process Mining, Event Data, XES Standard, Object-Centric Event Data, OCEL 2.0 Standard, Rust

1. Introduction

Apart from commercial process mining software, such as *Celonis* or *Fluxicon Disco*, there are also a few open-source solutions available, like ProM, PM4Py, or bupaR. The ProM framework, first presented in [1] is implemented in Java and features a graphical user interface (GUI) and a plugin system, with a large ecosystem of available plugins. ProM also has limited support for automated tasks through a command line interface (CLI). PM4Py is a Python software library and was first introduced in 2019 [2]. Instead of using a plugin system, users can implement custom approaches in Python programs that use the functionality exposed by PM4Py.

In [3], we presented an approach for implementing algorithms only once in Rust and exposing them via Java and Python bindings, making them available to both the PM4Py and ProM ecosystems, without any large re-implementation efforts. The initial implementation of prerequisites in Rust has since evolved into a standalone Rust library for process mining. In this paper, we introduce Rust4PM as a standalone process mining library project focused on performance. The project is available at https://github.com/aarkue/rust4pm.

We first describe the features of Rust4PM in Section 2. Next, in Section 3, we compare the performance of the different event data parsers implemented in ProM, PM4Py, and Rust4PM. To show its versatility, we present four example applications developed using Rust4PM in Section 4, focusing on different architectures and features. Finally, we conclude this paper in Section 5.

Proceedings of the Best BPM Dissertation Award, Doctoral Consortium, and Demonstrations & Resources Forum co-located with 22nd International Conference on Business Process Management (BPM 2024), Krakow, Poland, September 1st to 6th, 2024.

kuesters@pads.rwth-aachen.de (A. Küsters); wvdaalst@pads.rwth-aachen.de (W. M.P. van der Aalst)

0009-0006-9195-5380 (A. Küsters); 0000-0002-0955-6940 (W. M.P. van der Aalst)

2. Features

The main Rust4PM library is available at https://crates.io/crates/process_mining. It currently consists of four main modules, covering different functionalities.

- The `event_log` module contains the data structure definitions for the internal representation of traditional XES event logs. Additionally, it contains a full parser for the XES 2.0 standard, as well as export functionality for exporting event logs to XES. Moreover, there are case-streaming versions of the importer and exporter available, which allow processing cases of huge event logs without loading the full file in memory.
- The `ocel` module contains data structures for the OCEL 2.0 standard for object-centric event data, as well as import functionality for OCEL 2.0 files in the XML or JSON format.
- The `petri_net` module consists of data structures for Petri nets, as well as the ability to import and export basic Petri nets using the Petri Net Markup Language (PNML).
- The `alphappp` module implements the Alpha+++ process discovery algorithm, which was the starting point for the Rust implementation, as introduced in [3].

Moreover, Rust4PM integrates well with other projects. For example, PM4Py optionally supports importing XES or OCEL 2.0 files via the Rust4PM-based importer `rustxes` (see [3]).

3. Performance Evaluation

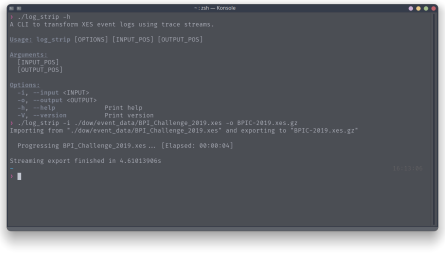
In this section, we present evaluation results for the performance of the XES and OCEL 2.0 XML parsers, as implemented in Rust4PM, PM4Py and (if available) ProM. For evaluation, we used a subset of the publicly available BPI Challenge XES event logs (see https://data.4tu.nl/search?search=BPI+Challenge) as well as OCEL 2.0 logs from https://www.ocel-standard.org/.

Figure 1: The total import durations for different XES and OCEL 2.0 files across the tested implementations. All configurations were repeated 5 times and the observed standard deviation is included as an error bar. Missing bars indicate a failed import (e.g., because the program ran out of memory).

The evaluation results are plotted in Figure 1. For all evaluated XES logs, the Rust4PM XES parser is at least two times faster on average than the other tested implementations. In comparison to PM4Py specifically, speedups of at least 10 times were observed for all evaluated XES event logs. For OCEL 2.0 XML, Rust4PM also significantly outperforms the PM4Py implementation by a factor of at least 15 for all evaluated files. As of now, no implementation of the new 2.0 version of the OCEL standard is available in ProM.

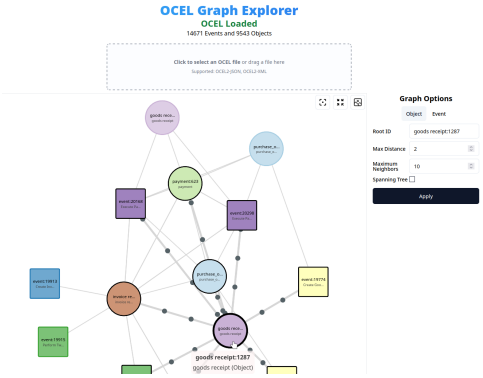
4. Use Cases: Example Projects and Architecture Recipes

In this section, we present four example applications built using the main Rust4PM library as a base. Figure 2 shows screenshots of the tools. Each application is based on a different architecture and usage context. Additionally, each application focuses on a specific feature set of the Rust4PM library. Thus, these examples do not follow a common theme, but should instead demonstrate the versatility of using Rust4PM in different contexts and architectures.

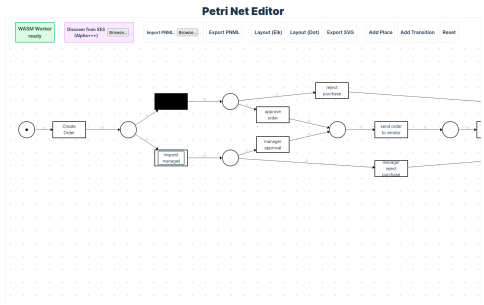


(a) `log_strip`

(b) `event_hours_analyzer`



(c) `ocel_graph`



(d) `petri_net_wasm`

Figure 2: The presented example applications using the Rust4PM library.

Next, we will shortly describe each demo application, mentioning the used architecture and leveraged library features. All examples are also publicly available at https://github.com/aarkue/rust4pm_demos, together with a *demo video* on Rust4PM as a whole, as well as these examples.

4.1. CLI for Stripping XES Attributes (`log_strip`)

The Rust command-line-interface (CLI) program `log_strip` can strip or modify certain attributes from an input XES event log, resulting in a stripped output log. A typical application for such a tool lays in privacy-preserving process mining, as presented in [4]. In particular, this tool removes all top-level log attributes, all case level attributes besides `concept:name` and all event-level attributes apart from `concept:name` and `time:timestamp`. Additionally, the second and nanosecond portion of the event timestamp is removed (i.e., set to 0), reducing the timestamp precision.

The tool uses the XES trace stream importer and exporter from Rust4PM. Thus, the CLI can also be used to modify event logs which are too large to fit in system memory.

4.2. Python Script for Data Visualization (`event_hours_analyzer`)

`event_hours_analyzer` is a Python Jupyter notebook for visualizing the number of events per hour of the day based on an input event log. The computationally expensive parts, i.e., the import of the XES event log and the computation of the event counts, are implemented in Rust. This functionality is then exposed as a Python library, which is used in the Jupyter notebook.

In this example, the XES importer and the event log data structures provided by the Rust4PM library are used.

4.3. Web Service for Constructing Graphs from OCED (`ocel_graph`)

This demo application is implemented as a web server backend implemented in Rust, which uses Rust4PM, and an interactive web-based frontend. The backend allows loading OCEL 2.0 event data and constructing graphs based on an event or object in the input data. In particular, all event-to-object and object-to-object relationships inside the object-centric event log are recursively expanded, based on specified parameters (e.g., the maximal number of recursion steps). The resulting interactive graph contains the encountered events and objects of the OCEL as nodes and edges between nodes corresponding to the relationships in the OCEL.

The backend utilizes the OCEL 2.0 XML or JSON import functionality of Rust4PM, as well as the OCEL 2.0 data structure representation to construct the graph.

4.4. Petri Net Editor with PNML and Discovery Support via WASM (`petri_net_wasm`)

This example application is a basic client-side Petri net editor. While the main editor is implemented using web technologies, it leverages the PNML import and export functionality from Rust4PM. Additionally, it allows discovering Petri nets from XES event logs based on the implementation of the Alpha+++ process discovery algorithm and the XES parser in Rust4PM. For that, the corresponding functions are exposed via WebAssembly (WASM), which is a possible compilation target for Rust. In contrast to the web server example presented before, WASM runs directly in the web browser of the user. We include this example specifically to demonstrate running Rust4PM in the web browser, which allows users to try out or use a tool simply by visiting a website, without the need to install anything on their machine.

5. Conclusion

In this paper, we introduced the performance-centric Rust4PM software project. Among other features, the main library supports importing the most common traditional and object-centric event data file formats, XES and OCEL 2.0. Through streaming XES traces, it also supports importing, processing and exporting huge XES event logs that would otherwise not fit into system memory. We evaluated the performance of the XES and OCEL 2.0 XML import parsers between two other popular open-source solutions, ProM and PM4Py, and observed significantly improvements in import speeds for all considered configurations, with speedups factors ranging from 2 to 40. To demonstrate the flexibility of Rust4PM, we presented four example applications, each with an own software architecture and feature focus.

Maturity The main Rust4PM library was first published in December 2023 with rudimentary features, and received more than 25 version updates since then. In this time, it was downloaded more than 10,000 times in total, according to crates.io. There are around 40 software test cases included, which cover a large portion of the implemented features. However, as the project is still rather young, there will likely still be changes to the exposed API surface in the future.

Future work There are still plenty of interesting features to be implemented in the library. For example, export support for OCEL 2.0, or the implementation of computationally expensive process mining algorithms, such as the computation of alignments. Furthermore, there are many possible advancements for the example applications presented in Section 4.

References

[1] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, W. M. P. van der Aalst, The ProM Framework: A New Era in Process Mining Tool Support, in: G. Ciardo, P. Darondeau (Eds.), Applications and Theory of Petri Nets 2005, 26th International Conference, ICATPN 2005, Miami, USA, June 20-25, 2005, Proceedings, volume 3536 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 444–454. URL: https://doi.org/10.1007/11494744_25. doi:10.1007/11494744_25.

[2] A. Berti, S. J. van Zelst, W. M. P. van der Aalst, Process Mining for Python (PM4Py): Bridging the Gap Between Process- and Data Science, CoRR abs/1905.06169 (2019). URL: http://arxiv.org/abs/1905.06169. arXiv:1905.06169.

[3] A. Küsters, W. M. P. van der Aalst, Developing a High-Performance Process Mining Library with Java and Python Bindings in Rust, CoRR abs/2401.14149 (2024). URL: https://doi.org/10.48550/arXiv.2401.14149. doi:10.48550/ARXIV.2401.14149. arXiv:2401.14149.

[4] M. Rafiei, W. M. P. van der Aalst, Privacy-Preserving Data Publishing in Process Mining, in: D. Fahland, C. Ghidini, J. Becker, M. Dumas (Eds.), Business Process Management Forum - BPM Forum 2020, Seville, Spain, September 13-18, 2020, Proceedings, volume 392 of *Lecture Notes in Business Information Processing*, Springer, 2020, pp. 122–138. URL: https://doi.org/10.1007/978-3-030-58638-6_8. doi:10.1007/978-3-030-58638-6_8.