The Performance Spectrum Miner: Visual Analytics for Fine-Grained Performance Analysis of Processes

Vadim Denisov[1], Elena Belkina, Dirk Fahland[1], and Wil M.P. van der Aalst[1,2]

[1] Eindhoven University of Technology, The Netherlands,
[2] Department of Computer Science, RWTH Aachen, Germany,
v.denisov@tue.nl, e.belkina@hotmail.com, d.fahland@tue.nl,
wvdaalst@pads.rwth-aachen.de

Abstract. We present the Performance Spectrum Miner, a ProM plugin, which implements a new technique for fine-grained performance analysis of processes. The technique uses the *performance spectrum* as a simple model, that maps all observed flows between two process steps together regarding their performance over time, and can be applied for event logs of any kinds of processes. The tool computes and visualizes performance spectra of processes, and provides rich functionality to explore various performance aspects. The demo is aimed to make process mining practitioners familiar with the technique and tool, and engage them into applying this tool for solving their daily process mining-related tasks.

Keywords: process mining, performance analysis, performance spectrum

1 Introduction

Process mining brings together traditional model-based process analysis and data-centric analysis techniques by using event data to obtain process-related information [2] for various goals, for example, answering performance-oriented questions [1]. Performance analysis is an important element in process management relying on precise knowledge about actual process behavior and performance to enable improvements [4]. Within process mining, performance analysis is one of the main types of model-based analysis of business processes, it is typically focused on performance indicators of the time dimension, such as the lead-, service- and waiting time and, as the name implies, is based on a process model. Many commercial and free process mining tools allow to do such analysis[3]. Despite all the benefits, model-based performance analysis has two significant drawbacks: 1) the commonly used model notations are not designed to project the time dimension on the model, i.e. changes over time cannot be represented in a comprehensible way and 2) process performance is always distorted by projection to a model, because no ideal models exist. The latter can be unacceptable for performance problems investigations, where inaccuracy in the obtained performance information may lead to wrong conclusions. Performance analysis based on models is limited, *Dotted Chart* [5] shows seasonal patterns and arrival rates, but no details on performance of process steps. Recently introduced performance spectrum [3] maps all observed

[3] For example, the ProM framework and Fluxicon Disco allow such analysis.

F. Casati et al. (Eds.): Proceedings of the Dissertation Award and Demonstration, Industrial Track at BPM 2018, CEUR-WS.org, 2018.

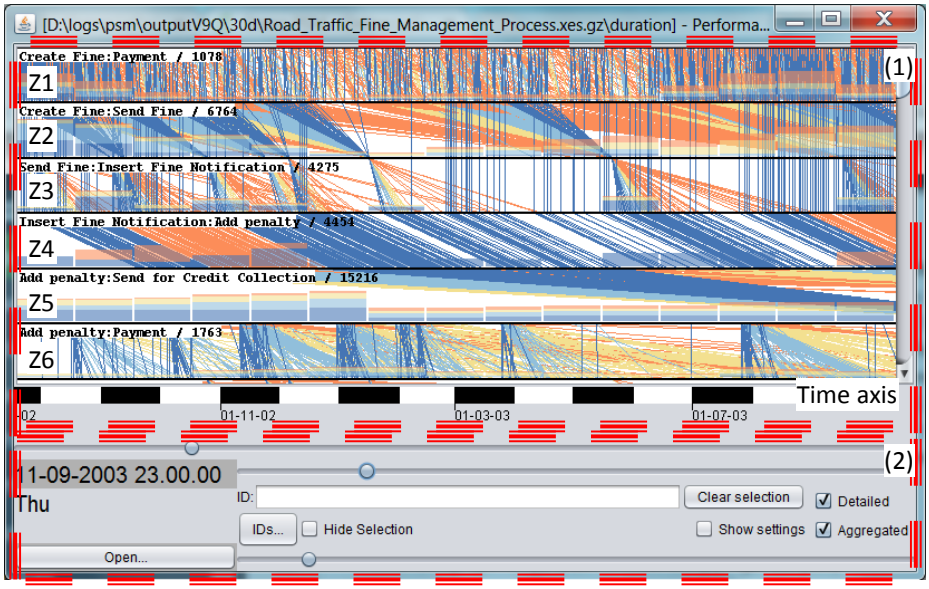


Fig. 1. The main window of the PSM with the main panel (1) and control panel (2).

flows between two process steps together regarding their performance over time. Our tool generates performance spectra of processes, assigns a class to each observed flow between two process steps (*segments*), according to a chosen *performance classifier*, *samples* the obtained data into *bins*, *aggregates* the data in bins and *visualizes* all the data *over time*. A user can explore a process performance spectrum by showing and hiding its *detailed* (i.e. non-aggregated) and *aggregated* parts, by scrolling and zooming, by filtering, aggregating and sorting segments, searching and highlighting required pieces of performance spectrum elements and so on, thereby enabling process mining practitioners with a new approach for performance analysis. The rest of this work is organized as follows. In Sect. 2, we explain a concept of the performance spectrum by example, in Sect. 3 we review the tool architecture, followed by extracts from our tool evaluation in Sect. 4, including scalability aspects of the PSM.

2 Tool

The tool has been developed as an interactive ProM plugin the Performance Spectrum Miner (PSM) in package "Performance Spectrum"[4] with an option to run as a stand-alone desktop application. In the remainder, we focus on key functionality of the PSM[5].

The main windows of the PSM is shown in Fig. 1. It consists of two parts: the scrollable *main panel* (1) and the *control panel* (2). During an analysis session in the PSM, a user first imports and pre-processes an event log, providing pre-processing

[4] source code available at `https://github.com/processmining-in-logistics/psm`

[5] watch a brief introduction to the PSM here: `https://www.dropbox.com/sh/yz214lpasw5ovu8/AABORHjYQdDbPCRS_-KyfAA1a?dl=0`

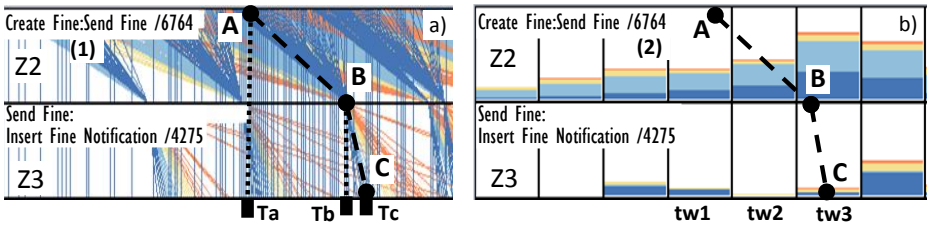


Fig. 2. Examples of a *detailed* (a) and *aggregated* (b) performance spectrum.

parameters, which are explained further in this section, then analyzes an obtained performance spectrum in the main panel. A performance spectrum consists of segments, that represent observed flows between two process steps over the time axis. It can be detailed, aggregated or combined. A detailed performance spectrum shows information about individual traces. For instance, in Fig. 2 a) segment Z2 represents a step between activities *Create Fine* and *Send Fine*, and has name *Create Fine:Send Fine*. Each spectrum line within the segment, e.g. highlighted line AB, represents occurrences of *Create Fine* that are followed by *Send Fine*. Occurrences of activities in points A and B have timestamps Ta and Tb correspondingly. Similarly, within Z3, line BC represents a case that has activity *Send Fine*, which is directly followed by activity *Insert Fine Notification*, which has timestamp Tc. Angles of lines indicate duration of steps: vertical lines show instant execution, while sloping lines indicate slower execution. The colors of lines show performance classes, assigned by a selected classifier. Available classifiers and the legend for the colors are shown in Fig. 4. While a detailed performance spectrum provides insight about individual cases, it does not directly visualize any quantified information. Therefore an aggregated performance spectrum serves for that purpose: within it, segments are split vertically into time windows, or bins, of a given duration, as shown in Fig. 2 b). Each bin contains a histogram that shows aggregated information about lines of the detailed performance spectrum that start, stop or intersect this bin. Besides the histograms, exact numbers are also available for users. Supported aggregation functions are presented in Fig. 3. In Fig. 2 b) bars in bins show aggregation by *cases pending* function. For instance, line AB is counted within corresponding *dark blue* bars (i.e. for class *0-25%*) in time windows tw1-tw3 of Z2. Additionally, parameter *maximal observed throughput* is shown within each segment (see Fig. 2 b) (2)). It shows the maximal observed value of the aggregation function within bins of the segment. The size of time windows, performance classifier and aggregation function are configured before pre-processing of an event log.

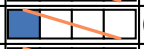
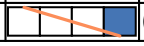
Aggregation function	Example	Result for bins
cases pending		(1, 1, 1, 1)
cases started		(1, 0, 0, 0)
cases stopped		(0, 0, 0, 1)

Fig. 3. *Aggregation functions.*

Classifier	Blue	Light-blue	Yellow	Orange
Quartile-based	0-25%	26-50%	51-75%	76-100%
Median-based	< 1.5*median	< 2*median	< 3*median	>= 3*median

Fig. 4. Available in the PSM *performance classifiers* and their color codes.

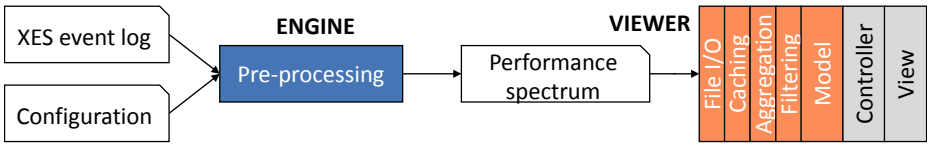


Fig. 5. The decoupled architecture of the PSM: the Engine and Viewer interacting via files on disk.

3 Architecture

The PSM architecture consists of two decoupled parts, as shown in Fig. 5: the *pre-processing engine* and the *viewer*. The engine processes an event log, represented in memory as an *OpenXES* `XLog` object, computes a process performance spectrum, using user-defined parameters, and export it to disk. An exported performance spectrum consist of two sets of files: one set contains bins with the aggregated performance information, and another one contains classified traces of the initial event log, which are stored on disk in a way efficient for *load-on-demand*. The *aggregation function* and *performance classifier* are selected by a user before the pre-processing step. The viewer has a traditional *model-view-controller* architecture, where the model serves as a data-source that hides many implementation details, such as a data storage type, file formats, a caching strategy, segments aggregation, filtering and sorting. The controller implements the business logic of the viewer, using high-level APIs of the model and view. Export of a computed performance spectrum to disk allows to avoid repetitions of the event log pre-processing phase for every session of analysis and decouples the engine and viewer. The engine, model and controller are implemented in the *Scala* programming language and based on the *Scala collections*, which allow extremely compact readable code and enable utilization of multi-core hardware architectures out of the box. The chosen architecture allows to replace easily an implementation of the engine, model or GUI without touching other components, for example, for switching to a high-performance storage or another pre-processing algorithm that takes some domain-specific event attributes into account.

4 Interactive Exploration of Performance Spectra

Here we focus on interactive features, evaluation and scalability aspects of the PSM. A user has a rich toolset to explore a performance spectrum: 1) regular expression based filtering of segments by names, 2) filtering by throughput boundaries, 3) searching for traces in a performance spectrum by specifying their IDs, 4) providing various segment sorting orders. Additionally, a user can *filter in* particular performance classes, for instance, compare the spectrum in Fig. 6 a), where only segments of classes *51-75%* and *76-100%* are shown, with the original spectrum in Fig. 2 a). Another feature of the PSM allows to highlight all segments of cases that in the performance spectrum have lines *that start in particular bins*. For instance, in Fig. 6 b) by selecting bin tw3 we highlight traces inside triangles ABC, CDE: they form a clearly distinguishable "hourglass" pattern within Z2-Z3, which shows that the traces are synchronized by activity *Send Fine* in point C. Interestingly, in Fig. 1 we observe more "hourglass" patterns within Z2-Z3, together with other patterns, for example, strictly parallel lines of Z4 or spreading lines of Z6. By

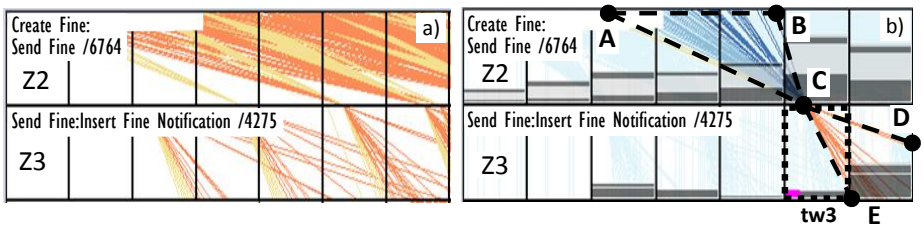


Fig. 6. The PSM features: filtering by performance classes (a) (see the original spectrum in Fig. 2 a)), and highlighting traces that have segments starting within selected bin tw3 (b).

default the PSM sorts segments alphabetically, and to work with multi-segment patterns a user should sort them manually. Automatic sorting of segments is the subject of future work. Aforementioned features of the PSM allow to conduct extensive performance analysis of processes, including their performance patterns [3].

We applied our tool on 12 real-life event logs from business processes (BPI12, BPI14, BPI15(1-5), BPI17, BPI18, Hospital Billing, RF) and on one real-life log from a baggage handling system (BHS) provided by Vanderlande. We illustrated how the performance spectrum provides detailed insights into performance for RF; for BHS we report on a case study for identifying performance problems; and we summarize performance characteristics of the 11 business process logs. Our analysis revealed a large variety of distinct patterns of process performance, which we organized into a taxonomy. We refer to [3] for discussion of the results.

Scalability of the PSM is different for its components. Applicability of the *engine* is limited by amount of RAM available for representation of an event log together with its performance spectrum. The required amount of RAM is proportional to an initial event log size and a chosen number of bins. On average a log with 1.000.000 events can be easily processed on a laptop with 16Gb of RAM. The *viewer* in the *load-on-demand* mode requires as little as amount of memory required for representation of one bin of each segment and allows to work with huge event logs (>10.000.000 events) on laptops with 16Gb of RAM. A faster *all-in-memory* mode requires roughly the same amount of memory as the engine. The engine's limitations can be eliminated by switching to a big-data platform, e.g. the Apache Spark, and the viewer's performance in the load-on-demand mode can be increased by moving to a high-performance data storage.

References

1. Process mining in practice. `http://processminingbook.com/`, accessed: 2018-06-04
2. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
3. Denisov, V., Fahland, D., van der Aalst, W.M.P.: Unbiased, fine-grained description of processes performance from event data. In: BPM 2018. LNCS, Springer (2018)
4. Maruster, L., van Beest, N.R.T.P.: Redesigning business processes: a methodology based on simulation and process mining techniques. Knowl. Inf. Syst. 21(3), 267–297 (2009)
5. Song, M., van der Aalst, W.M.: Supporting process mining by showing events at a glance. In: Proceedings of the 17th Annual Workshop on Information Technologies and Systems (WITS). pp. 139–145 (2007)