

CC4Spark: Distributing Event Logs and big complex Conformance Checking problems

Álvaro Valencia-Parra¹, Ángel Jesús Varela-Vaca¹, María Teresa Gómez-López¹ and Josep Carmona²

¹*Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Spain*

²*Universitat Politècnica de Catalunya, Spain*

Abstract

Conformance checking is one of the disciplines that best exposes the power of process mining, since it allows detecting anomalies and deviations in business processes, helping to assess and improve the quality of these. This is an indispensable task, especially in Big Data environments where large amounts of data are generated, and where the complexity of the processes is increasing. CC4Spark enables companies to face this challenging scenario in twofold. First, it supports distributing conformance checking alignment problems by means of a Big Data infrastructure based on Apache Spark, allowing users to import, transform and prepare event logs stored in distributed data sources, and solve them in a distributed environment. Secondly, this tool supports decomposed Petri nets. This helps to noticeably reduce the complexity of the models. Both characteristics help companies in facing increasingly frequent scenarios with large amounts of logs with highly complex business processes. CC4Spark is not tied to any particular conformance checking algorithm, so that users can employ customised algorithms.

Keywords

Conformance Checking, Big Data, Event Log Distribution

1. Introduction

It is increasingly common that organisations define complex business processes that must be followed to achieve their objectives [1]. These processes can include different kind of interactions with humans, third-party services, device's information, etc. Thereby these interactions can inadvertently enable unexpected deviations with respect to the expected process model. Further, this problem can be increased when the interactions come from multiple and heterogeneous sources, such as Internet-of-Things (IoT) devices or devices involved in Cyber-physical Systems [2]. Nowadays, conformance checking [3] techniques provide mechanisms to relate modelled and observed behaviour, so that the deviations between the footprints left by process executions and the process models that formalise the expected behaviour can be revealed.


Proceedings of the Demonstration & Resources Track, Best BPM Dissertation Award, and Doctoral Consortium at BPM 2021 co-located with the 19th International Conference on Business Process Management, BPM 2021, Rome, Italy, September 6-10, 2021

✉ avalencia@us.es (: Valencia-Parra); ajvarela@us.es (:J. Varela-Vaca); maytegomez@us.es (M. T. Gómez-López); jcarmona@upc.edu (J. Carmona)

🆔 0000-0002-0645-1000 (: Valencia-Parra); 0000-0001-9953-6005 (:J. Varela-Vaca); 0000-0002-3562-875X (M. T. Gómez-López); 0000-0001-9656-254X (J. Carmona)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

One of the major challenges in conformance checking is the *alignment problem*: given an observed trace σ , compute an end-to-end model run that more closely resembles σ . Computing alignments is an extremely difficult problem, with a complexity exponential in the size of the model or the trace [4]. Intuitively, computing an alignment requires a search through the state space of the model which, in certain cases implies an extensive exploration when the process model is large and/or highly concurrent. Traditionally, alignment problems are solved in a single node, which can increase the impact of the issue exposed above, especially in two situations: (i) when the dataset containing the event logs is too large, and (ii) when the model or Petri net is too complex. In those cases, calculating the alignment of a single trace can be very expensive in terms of computation time. In both cases, a solution could be distributing the alignment problems, solving them in parallel. As was demonstrated in [5], the impact in terms of computation time is reduced when the resolution of the alignment problems is distributed, since it tends to reduce the bottle-necks. Also, when Petri nets are too complex, the horizontal decomposition of them helps reduce that complexity.

We propose a tool to tackle the complexity of the alignment problem based on that principle. Thanks to the use of Apache Spark, CC4Spark can read distributed event logs, and solve alignment problems in distributed environments.

2. Innovation

CC4Spark¹ is the first tool that allows to distribute conformance checking alignment problems in big data clusters. This feature is supported by the use of Apache Spark. It offers a highly scalable way of processing both raw or XES event logs, and solving conformance checking alignment problems. This tool is the result of a previous study [5]. Next, the innovative features of this tool are presented:

Distributed event logs. CC4Spark is able to load raw process logs from distributed data sources, and to transform them into a XES-standard format. This feature allows the processing of event logs in parallel. CC4Spark is then capable of processing large volumes of event logs that would not be possible to process by a single machine. CC4Spark is also able to read non-distributed event logs contained in formatted XES files.

Processing horizontally-decomposed Petri nets. Horizontal decomposition of Petri nets is proposed in [5], and CC4Spark provides support for processing such models. Horizontal decomposition produces *partial models*, which are less complex than their original Petri nets due to their acyclicity. These partial models are generated by running multiple instances of the original Petri net. The set of partial models is equivalent to the original model. In this way, the complexity of cyclical Petri nets is noticeably reduced. In order to horizontally-decompose Petri nets, an external tool (VIP tool²) is required.

Solving distributed alignment problems. The alignment problem is distributed between the nodes of the cluster in partitions of *alignment subproblems*. Each subproblem consists of a specific trace and a Petri net (or a partial model if the Petri net is decomposed). In this way,

¹CC4Spark source code and documentation: <https://github.com/IDEA-Research-Group/conformancechecking4spark>, <http://www.idea.us.es/confcheckingbigdata/>

²VIP Tool: https://www.fernuni-hagen.de/sttp/forschung/vip_tool.shtml

all the possible combinations between different traces and Petri nets (or partial models) are generated and computed, generating *partial solutions* (i.e., a *candidate* solution composed of the trace and the best alignment found for that subproblem). The configuration of the partitions (i.e., the amount of partitions and the number of subproblems per partition) can be customised in this tool. CC4Spark also includes a heuristic to avoid computing unnecessary alignment subproblems. This heuristic consists of the estimation of a lower bound, indicating that the minimum value of alignment for that subproblem will never be lower than the estimated one. In that case, if the subproblems related to a certain trace produce a higher estimation than the last alignment computed for that trace, the rest of the subproblems related to that trace will be skipped. Finally, partial solutions are combined. The solution for each trace is the best alignment found for it.

Remark that CC4Spark is not tied to any conformance checking algorithm: user is free to use its own implementation. Up to now, two approaches have been successfully tested (i.e., the A* algorithm implemented in the Python library *PM4Py*, and an approach based on Constraint Optimisation Paradigm [5]).

3. Case Studies

Case Study A: Computing deviations on distributed event logs. This case study is related to the milk manufacturing industry. The process describes how milk cans are processed from when they leave the farm until they pass through the factory and are bottled for sale. Each machine in the process generates logs, which include the milk can being processed, the timestamp, and the values read by the sensors (e.g., temperature, pressure). The IoT infrastructure stores these logs in a MongoDB cluster. Then, CC4Spark reads such data. For the logs to be formatted in a XES format, the attributes which represent the case ID, the task ID, and the timestamp must be specified. Then, a path to a Petri net file must be specified. The results of the alignment problem will be store in the storage system specified by the user. Thanks to CC4Spark, it is possible to monitor the whole process and detect potential abnormalities in milk bottles.

Case Study B: Computing deviations with complex Petri nets. In this example, we present the evaluations performed in [5]. Several datasets are tested, each one with a specific XES event log file, and a Petri net. While the size of the event logs are less than in Example A, the Petri nets are highly complex. For this reason, in the distributed approaches, the Petri nets were decomposed by using the technique previously described. The results of the tests are depicted in Figure 1, comparing the best configuration obtained for the A* algorithm standalone (the classical approach, without distribution), the A* algorithm with the best distributed configuration, and a novel approach based on Constraint Programming for solving the alignment problems (refer to [5] for further information). The Elapsed Real Time depicted in the graph represent the average time of 10 executions. The results demonstrated that distributing the alignment problems and solving them in parallel improves the execution time. It is especially noticeable for the datasets *M2*, *M5*, *CCC20d*, and *prGm6* (in this case, neither of the other approaches were able to solve the alignment problems in a reasonable amount of time, being the A* algorithm in distributed mode the only capable of obtaining the results of the alignments).

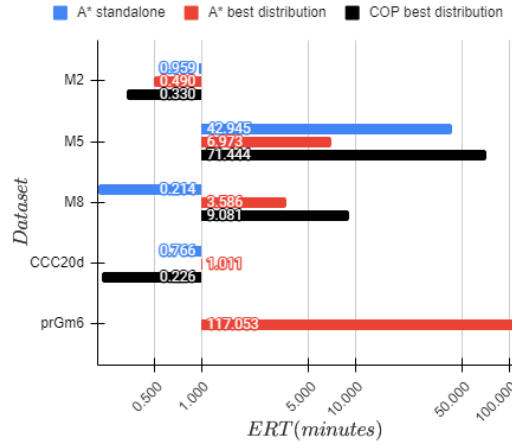


Figure 1: Comparison of the best configuration for each algorithm. The abscissa has a logarithmic scale

4. Tool Packaging and Maturity

CC4Spark is developed on top of PySpark 3.0.1. It requires Python 3.7 and PM4Py 2.2.8. It is packaged as a Python module which can be downloaded and imported in a PySpark project. Although this tool facilitates the process of importing event logs, Petri nets models, and calculating the alignments, certain skills in Apache Spark are required for a proper use of this tool. For this reason, in future releases we intend to extend this tool by including a graphical user interface. CC4Spark might be run standalone inside a PySpark project in a local computer. However, for the tool to properly work in distributed environments, a cluster based on Apache Spark is required. Our manuals include instructions on the deployment in both modes.

Figure 2 depicts the architecture when CC4Spark is deployed in distributed mode. There are three layers clearly differentiated: (i) Storage layer: It refers to the event log and Petri net models storage. As mentioned before, the event logs can be in distributed data sources or in a raw format. From the technical point of view, the storage layer might be any type of distributed database, distributed or local file system. (ii) Computing layer: It is intended to distribute the event logs into partitions, transform them into a XES format, generate the alignment subproblems, and solve them. CC4Spark relies on PySpark for this layer. (iii) Persistence layer: It stores the results of the alignment problems. It might be any database or storage system supported by PySpark.

5. Conclusion

CC4Spark enables users to perform conformance checking computations with large amounts of event logs and highly complex Petri nets. As reported in Section 3 and in [5], the scalability of CC4Spark is shown. It has been tested with distributed event logs, non-distributed event logs, standard Petri nets and decomposed Petri nets. In those cases, the tool has demonstrated scalability and an improvement with respect to the classical approaches when it runs in distributed

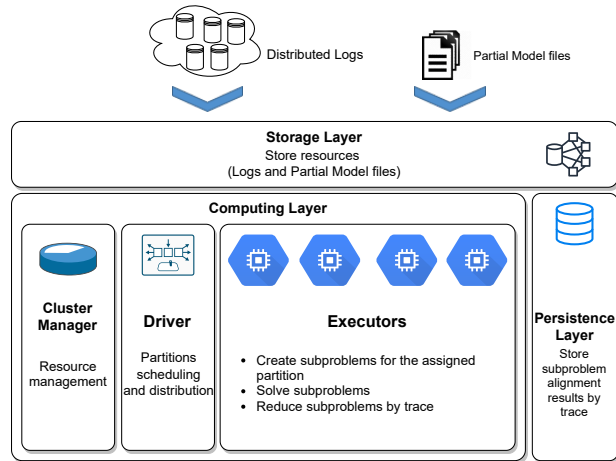


Figure 2: Architecture of CC4Spark

mode. A demonstration video is available³.

Acknowledgments

Ministry of Science and Technology of Spain: ECLIPSE (RTI2018-094283-B-C33) project; European Regional Development Fund (ERDF/FEDER); MINECO (TIN2017-86727-C2-1-R); University of Seville: *VI Plan Propio de Investigación y Transferencia* (VI PPIT-US).

References

- [1] M. Dumas, W. M. P. van der Aalst, A. H. M. ter Hofstede, *Process-aware Information Systems: Bridging People and Software through Process Technology*, Wiley, 2005. URL: <http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0471663069.html>.
- [2] H. Roehm, J. Oehlerking, M. Woehrle, M. Althoff, *Model conformance for cyber-physical systems: A survey*, *TCPS* 3 (2019) 30:1–30:26. URL: <https://doi.org/10.1145/3306157>. doi:10.1145/3306157.
- [3] J. Carmona, B. F. van Dongen, A. Solti, M. Weidlich, *Conformance Checking - Relating Processes and Models*, Springer, 2018. URL: <https://doi.org/10.1007/978-3-319-99414-7>. doi:10.1007/978-3-319-99414-7.
- [4] A. Adriansyah, *Aligning observed and modeled behavior*, Ph.D. thesis, Technische Universiteit Eindhoven, 2014.
- [5] Á. Valencia-Parra, Á. J. Varela-Vaca, M. T. Gómez-López, J. Carmona, R. Bergenthum, *Empowering conformance checking using Big Data through horizontal decomposition*, *Information Systems* 99 (2021) 101731. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0306437921000077>. doi:10.1016/j.is.2021.101731.

³Demo video: <http://tiny.cc/cc4spark-video>