Handling Big(ger) Logs: Connecting ProM 6 to Apache Hadoop

Sergio Hernández[1], S.J. van Zelst[2], Joaquín Ezpeleta[1], and Wil M.P. van der Aalst[2]

[1] Department of Computer Science and Systems Engineering
University of Zaragoza, Spain
{shernandez, ezpeleta}@unizar.es
[2] Department of Mathematics and Computer Science
Eindhoven University of Technology, The Netherlands
{s.j.v.zelst, w.m.p.v.d.aalst}@tue.nl

Abstract. Within process mining the main goal is to support the analysis, improvement and apprehension of business processes. Numerous process mining techniques have been developed with that purpose. The majority of these techniques use conventional computation models and do not apply novel scalable and distributed techniques. In this paper we present an integrative framework connecting the process mining framework ProM with the distributed computing environment Apache Hadoop. The integration allows for the execution of MapReduce jobs on any Apache Hadoop cluster enabling practitioners and researchers to explore and develop scalable and distributed process mining approaches. Thus, the new approach enables the application of different process mining techniques to events logs of several hundreds of gigabytes.

Keywords: Process mining, Big Data, scalability, distributed computing, ProM, Apache Hadoop

1 Introduction

We assume the reader to be knowledgeable with regard to the basics of process mining and refer to [1] for an in-depth overview.

Nowadays, we are able to store huge quantities of event data. In principle, an array of process mining techniques can be used to analyse these data. However, classical process mining techniques are not able to cope with huge quantities of data. Within the ProM framework[3] [2] it is currently impossible to analyse event data whose size exceeds the computer's physical memory. Within process mining only a limited amount of research has been done concerning the integration of techniques that are designed to cope with enormous amounts of data. *Divide and conquer* based approaches have been developed in order to reduce computational complexity [3]. Some work has been done to incorporate stream mining techniques within the context of process mining [4,5], i.e. the input data is regarded as a stream of events rather than a static event log. Finally, the application of MapReduce techniques to process discovery is explored in [6].

[3]http://www.promtools.org/

Apache Hadoop[4] provides an open-source multi-purpose framework which main aim is to provide reliable, scalable and distributed computing. Typically, Hadoop runs on a large-scale computational cluster of servers. It comprises of a set of different modules that handle different perspectives of the aforementioned aim. The *Hadoop Distributed File System* (HDFS) component provides distributed storage whereas the *Yet Another Resource Negotiator* (YARN) component implements a MapReduce programming model aimed at processing vast amounts of data. In particular, MapReduce finds its fundamental concepts within the area of functional programming and is particularly aimed at handling large amounts of semi- and/or unstructured data.

Although some interesting results regarding the use of MapReduce in process mining are shown in [6], the techniques are not available to the process mining community. Thus, a unifying implementation that allows the development of scalable and distributed process mining techniques does not yet exist. Moreover, a thorough investigation of techniques related to distributed computation models for process mining is missing. In this paper, we present a newly developed framework integrating the process mining framework ProM and Apache Hadoop. The integration allows any user of the ProM framework to execute MapReduce jobs on any given Apache Hadoop cluster. Thereby, the framework is intended to provide an easy entry point for the development of MapReduce-based techniques from a process mining perspective.

The remainder of this paper is organized as follows. In Section 2 we present the core concepts of the newly created integrative framework. Section 3 shows an example of execution of a process discovery based MapReduce job within Hadoop using the framework. Section 4 concludes the paper.

Additionally, a screencast detailing the operation of the developed plugins and showing an use case is provided here: https://svn.win.tue.nl/repos/prom/Packages/Hadoop/Tags/publications/screen_casts/2015_bpm_demo_hadoop/hadoop_demo_2015.tar.gz

2 Core Concepts

The main purpose of the integration between Apache Hadoop and ProM is to enable researchers and practitioners to use, develop and/or publish Hadoop-based process mining techniques. Hence, the newly developed framework acts as a core platform that establishes access to a variety of Apache Hadoop based functionality. Conceptually, the basic goal of the integration is to enable any user and/or developer to connect their Apache Hadoop cluster[5] to ProM and design/execute MapReduce-based process mining tasks.

The `HadoopClusterParameters` (`HCP`) interface acts as the core of the integration between Apache Hadoop and the ProM framework by managing the connection to a Hadoop cluster within ProM. The `HCP` object is needed by all Hadoop plugins. Currently, the `HCP` object provides means to verify whether a connection can be established to the Hadoop cluster, by verifying whether the HDFS can be mounted and the user has access to the cluster. A second core element of the integration is the `HDFSXLog` interface which extends the well-known `XLog` interface[6]. As Apache Hadoop provides a

[4]http://hadoop.apache.org/

[5]Given that the Hadoop cluster fits a specific range of Hadoop release lines, e.g. 2.x.y.

[6]http://www.xes-standard.org/openxes/start

distributed file system, i.e., HDFS, we can store (XES) event logs on the cluster. Within the Hadoop integration we provide means to *import* such event logs from the cluster. Importing an event log will create an `HDFSXLog` object within ProM. By default the log is not actually loaded in the local memory as it is rather a pointer to the external location of the event log. Note that currently, when importing the event log, it's size may not exceed the computer's physical memory.

By using the previous objects as input, different plugins implementing MapReduce-based process mining techniques could be used. Currently, we have implemented several process discovery techniques: the Alpha Miner [7], the Flexible Heuristics Miner [8], and the Inductive Miner [9]. To include new process mining techniques, the developer should focus on implementing the MapReduce-based application since the framework is able to manage the lifecycle of the applications executed in Hadoop. For that, the new technique must be packed into an executable Java Archive (jar) and it must be included within the ProM sources. Then, a new plugin executing the developed application can be created. The plugin can use the methods provided by the framework to manage the communication with the Hadoop cluster, transfer the corresponding JAR file, execute the MapReduce application and transfer back the results of the computation which can consequently be visualized by ProM.

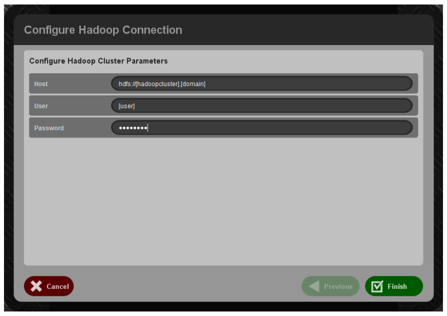
3 Case Study - Executing the Inductive Miner on Apache Hadoop

As a case study the execution of a newly developed plugin executing the Inductive Miner [9] in a Hadoop cluster is presented. The plugin computes the directly-follows graph used by the Inductive Miner in Hadoop and then gets and visualizes the final process model in ProM. The algorithm is applied on an event log whose size is *218 GB*.

Any plugin that is able to execute MapReduce jobs on a Hadoop cluster needs an `HCP` object as well as an `HDFSXLog` object. Importing an `HDFSXLog` can be done both with or without the use of an `HCP` object, i.e. we provide a plug-in that generates both objects. To import a `HDFSXLog` we run the "Import a XLog from Hadoop Distributed File System" plugin in ProM. If we start the plugin without using an `HCP`, the user is first asked to connect to a Hadoop cluster (Figure 1a). Next, the path to the specific log of choice must be provided (Figure 1b). Thus, the plugin generates two artifacts, one `HCP` object and one `HDFSXLog` object.

Using the latter two objects as an input we execute the "Mine a Process Tree in Hadoop using Inductive Miner (from DFG)" plugin. Executing this plugin triggers a copy of the associated jar files, i.e. specifying the MapReduce jobs, to the Hadoop cluster. If all files are transferred successfully the Hadoop job will be submitted. The progress of the job can be inspected within the Hadoop Cluster Metrics overview (which is part of the Apache Hadoop software) as depicted in Figure 2a.[7] The result of applying the Inductive Miner on the directly follows graph computed using Apache MapReduce is depicted in Figure 2b as a process tree.

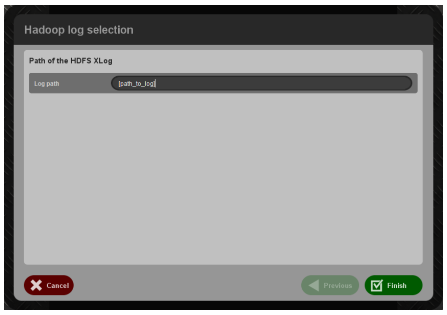
[7]Note that we removed some of the user and cluster specific information from the screenshot.


Configure Hadoop Connection

Configure Hadoop Cluster Parameters

Host	hdfs://[hadoopcluster].[domain]
User	[user]
Password	••••••••

(a) Screenshot of the dialog requesting the host, user and password for the `HadoopClusterParameters` (`HCP`).

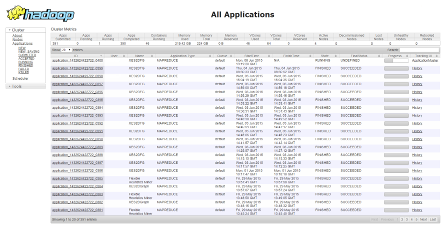

Hadoop log selection

Path of the HDFS XLog

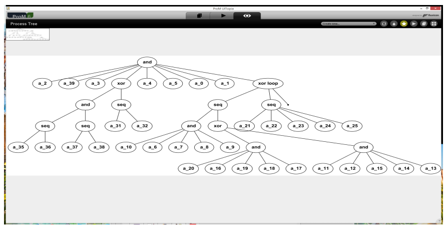
Log path	[path_to_log]

(b) Screenshot of the dialog requesting the path of the XES log in the HDFS to generate the `HDFSXLog` artifact.

Fig. 1: Screenshots of the dialogs shown by the "Import a XLog from Hadoop Distributed File System" plugin in ProM.



(a) Apache Hadoop cluster metrics.



(b) Inductive Miner Result.

Fig. 2: The Hadoop Cluster Metrics web interface showing the progress of the MapReduce jobs and the result of applying the Inductive Miner on a directly follows graph learned on an event log of 218 GB using Apache Hadoop.

4 Conclusion

The newly created integration between ProM and Apache Hadoop allows developers and BPM professionals to use and/or develop Big Data related techniques within a process mining context. The integration allows users and developers to connect, in a trivial manner, to an arbitrary Hadoop cluster. Thus, without any in-depth technical knowledge of Hadoop users can start exploring new types of analysis using a distributed and scalable computing infrastructure. Furthermore, this integration enables the analysis of huge event data logs.

Future Work There are many interesting directions for future work. Currently, the user needs to specify the exact path to a file. In the future we want to integrate support for browsing the HDFS using some graphical interface. Additionally we want to integrate user authentication more thoroughly throughout all plugins using the `HCP` object. Also, we want to develop a new importer for HDFS logs that reads the log as a stream. In this way we could actually import logs which size exceeds the computer's physical memory allowing the use of all ProM plugins with large event logs.

Another interesting addition is "on-the-fly" jar generation. Currently, when developing a plug-in that (partially) consists of MapReduce tasks, the developer needs to manually generate a jar file and include it in the project source. We think of extending the framework in such way that the jars needed for the execution of MapReduce on Hadoop will be automatically generated within ProM. This allows developers to primarily focus on the implementation of MapReduce related code and abstract from the administrative details of handling the execution on a cluster.

Acknowledgment This work has been supported by the research projects 287230/2, granted by Gobierno de Aragón, FPU12/04775, granted by the Spanish Ministerio de Educación, Cultura y Deporte, TIN2014-56633-C3-2-R, granted by the Spanish Ministerio de Economía y Competitividad, and the DELIBIDA (Desire Lines in Big Data) research program supported by NWO.

References

1. Aalst, W.M.P.v.d.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. 1st edn. Springer Publishing Company, Incorporated (2011)
2. Dongen, B.F.v., Alves de Medeiros, A.K., Verbeek, H.M.W., Weijters, A.J.M.M., Aalst, W.M.P.v.d.: The prom framework: A new era in process mining tool support. In Ciardo, G., Darondeau, P., eds.: Applications and Theory of Petri Nets 2005. Volume 3536 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2005) 444–454
3. Aalst, W.M.P.v.d.: Decomposing Petri nets for process mining: A generic approach. Distributed and Parallel Databases **31**(4) (2013) 471–507
4. Burattin, A., Sperduti, A., Aalst, W.M.P.v.d.: Control-flow discovery from event streams. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China, July 6-11, 2014. (2014) 2420–2427
5. Zelst, S.J.v., Burattin, A., Dongen, B.F.v., Verbeek, H.M.W.: Data streams in ProM 6: A single-node architecture. In: Proceedings of the BPM Demo Sessions 2014 Co-located with the 12th International Conference on Business Process Management (BPM 2014), Eindhoven, The Netherlands, September 10, 2014. (2014) 81
6. Evermann, J.: Scalable process discovery using Map-Reduce. Services Computing, IEEE Transactions on (2014)
7. Aalst, W.M.P.v.d., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. Knowledge and Data Engineering, IEEE Transactions on **16**(9) (2004) 1128–1142
8. Weijters, A.J.M.M., Ribeiro, J.T.S.: Flexible Heuristics Miner (FHM). In: Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on, IEEE (2011) 310–317
9. Leemans, S.J.J., Fahland, D., Aalst, W.M.P.v.d.: Discovering block-structured process models from event logs - A constructive approach. In: Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings. (2013) 311–329