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

Sergio Hernàndez[1] and S.J. van Zelst[2]

[1] Department of Computer Science and Systems Engineering
University of Zaragoza, Spain
shernandez@unizar.es
[2] Department of Mathematics and Computer Science
Eindhoven University of Technology, The Netherlands
s.j.v.zelst@tue.nl

Abstract. Within process mining the main goal is to support the analysis, improvement and apprehension of business processes. A vast amount of techniques has been developed within the field. The majority of these techniques however comprise of a rather classical computational fashion and do not incorporate the latest developments aimed at tackling the information explosion phenomenon, e.g. techniques related to scalability and distributed computing. In this paper we present a newly developed 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 new work related to scalable and distributed computation, from a process mining perspective.

Keywords: Process mining, big data, scalability, distributed computing, ProM, Apache Hadoop

1 Introduction

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

Nowadays we are able to store huge quantities of data related to business process execution. Classical process mining techniques however are simply not able to cope with these quantities of data, i.e. within the ProM framework [2, 3][3] it is currently impossible to analyse event data which 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. W.r.t. event log based process discovery, techniques have been developed to apply *divide and conquer* based approaches in order to reduce computational complexity [4]. Some work has been done w.r.t. the incorporation stream mining techniques within the context of process mining [5–7], i.e. the input data is regarded as a stream of events rather than a

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

static event log. Finally, some work has been conducted w.r.t. the application of MapReduce techniques to process discovery [8].

Apache Hadoop[4] provides open-source multi-purpose software 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 a distributed file system whereas the MapReduce component implements a 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 big amounts of semi- and/or unstructured data.

Although [8] shows some interesting results that hint on a thorough investigation of techniques related to distributed computation models w.r.t. process mining, a unifying implementation that allows development of prototypes of the like does not yet exit. 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 executed MapReduce jobs on any given Apache Hadoop cluster. Moreover the framework is intended to provide an easy entry point for the development 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 presents an example of execution of a process discovery based MapReduce job within Hadoop using the framework. Section 4 concludes the paper.

2 Core Concepts

The main purpose of the integration between Apache Hadoop and ProM is to enable researchers, practitioners etc. 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.

At the core of the integration between Apache Hadoop and ProM acts the `HadoopClusterParameters` (`HCP`) interface, specifying the connection 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 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.

[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`

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. However, it can also be used in the standard ProM plugins. In this situation, the log is directly read from the HDFS and loaded in the local computer's memory. Note that currently, when actually importing the event log, it's size may not exceed the computer's physical memory.

The main architectural challenge within establishing the connection between ProM and Hadoop is the actual execution of a Hadoop MapReduce job. As the event data under study, possibly too large to store on a regular computer, is stored in the HDFS component of the Hadoop cluster, all computation should preferably be performed on the cluster rather than on the local machine. This implies that as a prerequisite for executing a specific MapReduce job, the corresponding implementing code (which is programmed using the JAVA language) needs to be packed into a Java Archive (jar) and send to the Hadoop cluster. From a developer's perspective this means that after developing a MapReduce job, the corresponding JAR file is to be included within the ProM sources. Then, the framework provides methods to transfer the executable JAR, execute the MapReduce job and transfer back the results of the computation which can consequently be visualized by ProM.

3 Case Study - Executing the Alpha Miner on Apache Hadoop

As a case study we have implemented calculation of a directly follows graph as a set of MapReduce tasks. Te directly follows graph can be used as an input for the Inductive Miner [9]. We applied it on a event log which total 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 as depicted in Figure 1a. The plugin will generate two artifacts, one `HCP` object and one `HDFSXLog` object. After specifying a connection, the user is prompted to provide a path to the specific log of choice as depicted in Figure 1b.

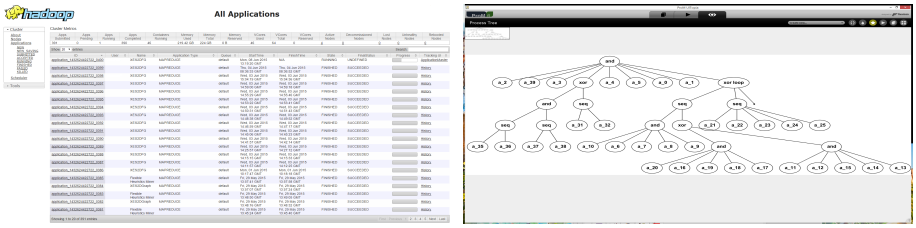
If the user has specified the path to the event log and clicked the finish button the `HCP` object and the `HDFSXLog` will be created. Using the latter two objects as an input we execute the "Mine a Process Tree in Hadoop using Inductive Miner (from DFG)" plugin. Executing this plugin triggers a copy 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 started. The progress of the job can be inspected within the Hadoop Cluster Metrics overview (which is part of 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.

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

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

(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 218GB using Apache MapReduce.

4 Conclusion

The newly created integration between ProM and Apache Hadoop allows developers, BPM professionals, etc. 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 in data intensive environments.

Future Work We identify several 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 can import logs which size exceeds the computer's physical memory.

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.

References

1. Aalst, W.v.d.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. 1st edn. Springer Publishing Company, Incorporated (2011)
2. Dongen, B.v., Alves de Medeiros, A., Verbeek, H., Weijters, A., Aalst, W.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.v.d., Dongen, B.v., Günther, C., Mans, R., Alves de Medeiros, A., Rozinat, A., Rubin, V., Song, M., Verbeek, H., Weijters, A.: Prom 4.0: Comprehensive support for real process analysis. In Kleijn, J., Yakovlev, A., eds.: Petri Nets and Other Models of Concurrency – ICATPN 2007. Volume 4546 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2007) 484–494
4. Aalst, W.M.P.v.d.: Decomposing petri nets for process mining: A generic approach. Distributed and Parallel Databases **31**(4) (2013) 471–507
5. Maggi, F.M., Burattin, A., Cimitile, M., Sperduti, A.: Online process discovery to detect concept drifts in ltl-based declarative process models. In: On the Move to Meaningful Internet Systems: OTM 2013 Conferences - Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings. (2013) 94–111
6. Burattin, A., Sperduti, A., Aalst Wil, 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
7. 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
8. Evermann, J.: Scalable process discovery using map-reduce. Services Computing, IEEE Transactions on **PP**(99) (2014) 1–1
9. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: 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