Process Discovery using Big Data stack - Implementing the Alpha Algorithm with Map-Reduce

Design Specification Document

Martin Hashem, Xiangan Chen

May 13, 2019
RWTH Aachen

1 Introduction

Process mining is an approach to extract process models from event logs. Since the distributed nature of modern information systems, event logs are likely to be distributed among different physical machines. Map-Reduce is a scalable approach for efficient computations on distributed data. In this Python application we will present the main idea of a Map-Reduce implementation of the Alpha process mining algorithm, to take advantage of the scalability of the Map-Reduce approach.

To fulfill the shortcoming in the current process mining technology, this project aims to build a new python-based web service, that integrates the big data capabilities of the Hadoop system into the process mining framework pm4py.

This document is presented for our shareholders of the project.
Within our project we mostly apply well-established open source applications, which includes the following tools with version numbers:

- Python 3.6 as back end programming language
- pm4py 1.1.10 as process mining toolkit
- Flask 1.0.2 as web framework
- Hadoop 3.2.0 as big data processing and distributed computation framework
- Docker CE as deploy platform

This very document is one of these, which will be provided to bring insights to our software development cycle:

- Project initiation document
- Requirements analysis document
- Design specification document
- Software documentation

2 System Overview

The main goal of this project is to create a web application that provides calculations with the Alpha algorithm and can prepare the data using the Map-Reduce.

The web service is to be accessed by a standard browser and needs to provide the following functionalities:

- Upload event logs in either CSV or XES format into the Hadoop system
- Front end links to existing algorithms for further processing
- Do Map-Reduce computations to run the alpha algorithm
- Download the files onto local system

3 Design Considerations

3.1 Assumptions and Dependencies

For an easy deployment for our entire system, we decided to use Docker, since it also is used for the distributed filesystem Hadoop. This Module will also be conform with PM4py and will share an interface with it. For this integration the programming language will be Python and for the webframework we choose the minimal framework Flask, for a resource friendly environment.

The standard setup will be a single system Hadoop node, but it can be easily changed to multiple (existing) nodes. A deploy script will be available for this in Linux (Debian, Ubuntu will be supported for sure) , and as parameters only the ip's / addresses of the nodes will be neccessary [this might be subject to change].

3.2 General Constraints

The deployment is set for a server of unknown size, so a general minimum will be set, and tested for the deploy. To be integratable into PM4py the language for front and back end will be Python. The front end will needed to be accessibil from a standard internet browser and multiple instances can be expected, therefore minimal framework is needed. The server backend will need to calculate the MapReduce for the eventlogs and then use the Alpha-Algorithm on the MapReduce logs. The outputted petri net is needed to be ready for further access in PM4py. It needs to be downloadable from the frontend as well.

3.3 Goals and Guidelines

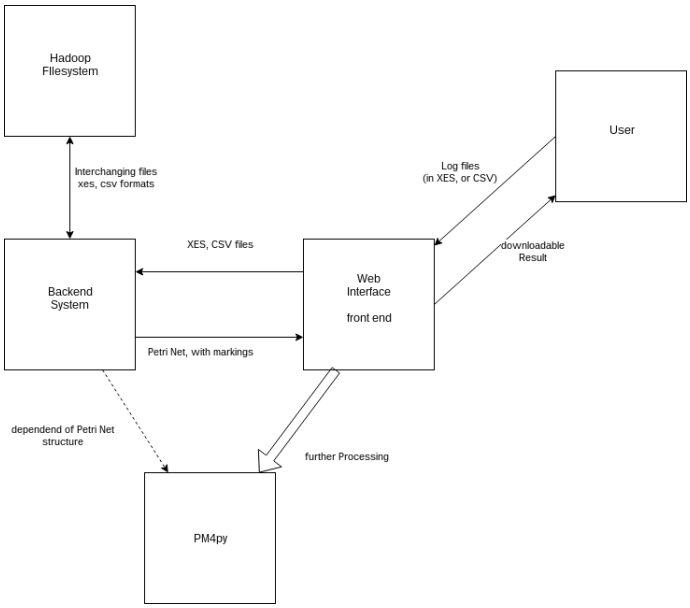
- **Simple:** One of the main guidelines through our system design is simple. From the frontend part to the backend, from the UIs to the layout, we set the simpleness on the crucial level.

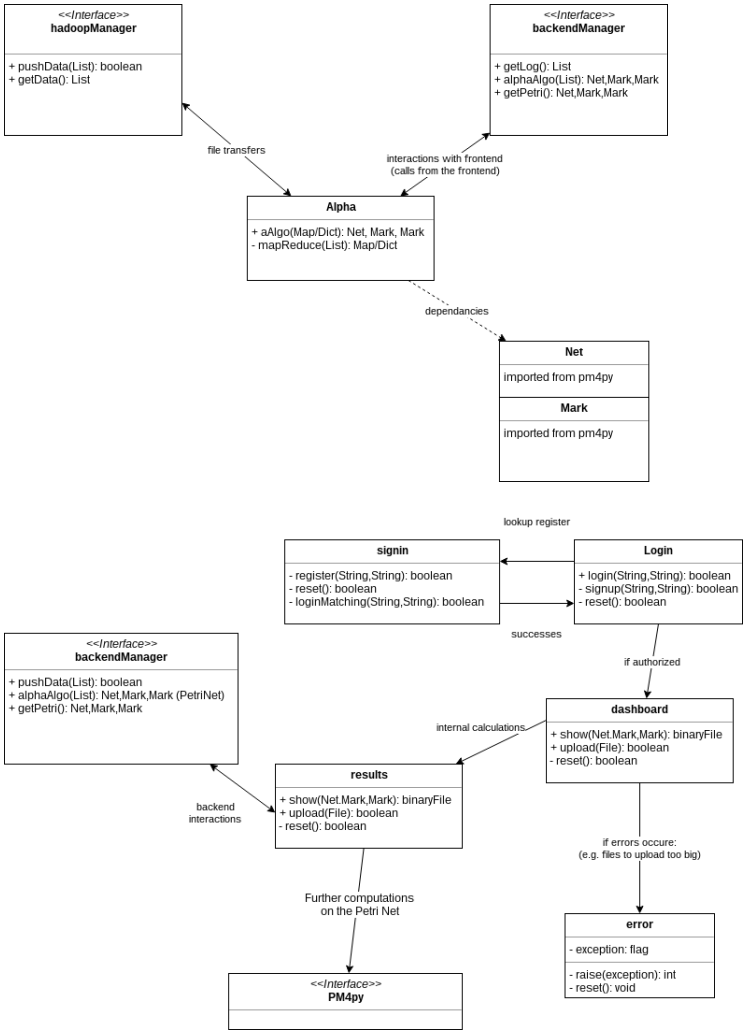
- **Resource-saving:** The other goal of our design is resource-saving. Not only in our implement but also the outcoming software, we consume resources as little as possible, inclusive CPU computing time , storage and server occupancy.

3.4 Development Methods

Scrum is our first choice of development method. Since we have a small team, so an agile and intensive development method should fit us better. So instead of the classic waterfall model development, our team prefer Scrum. Scrum is a lightweight, iterative and incremental framework for managing product development. And enables our team to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members.[**?**]

4 System Architecture





5 Policies and Tactics

6 Detailed System Design

6.1 login

Name: signIn
Returns: -
Arguments: -
Description: links to the signIn page to let new user to register.

Name: login

Returns: -
Arguments: username(String), password(String)
Description: with the previously registered userdata the user can login to the index page (dashboard).

Name: reset
Returns: -
Arguments: -
Description: clears all filled items of the login page.

6.2 signIn

Name: register
Returns: Boolean
Arguments: username(String), password(String), email(String).
Description: lets new user to register, and returns Ture when all information are valid.

Name: reset
Returns: -
Arguments: -
Description: clears all filled items of the signIn page.

6.3 dashboard

Name: upload
Returns: Boolean
Arguments: File
Description: uploads the from user selected file, will return a boolean of success.

Name: reset
Returns: -
Arguments: -
Description: deletes the uploaded file and ready to recieve a new one.

Name: proceed
Returns: Boolean
Arguments: File
Description: tests if the uploaded file is valid, if so it will transfer the file to backend system,and return a True.

6.4 result

Name: show
Returns: Graph(svg)
Arguments: Net, initMark, finMark
Description: gets the result from alpha algorithm calculation and presents the result of the processed file in petri-net.

Name: download
Returns: -
Arguments: Graph(svg)
Description: allows user to download the graph of proceeded file onto local system.

Name: furtherProcess
Returns: -
Arguments: Net, initMark, finMark
Description: transfers the raw file from alpha calculation to PM4py for further process.

6.5 Net

Name: Net
Type: Class
Attributes: places, transitions, arcs
Description: Inherited Petri Nets from PM4py for further use in PM4py.

6.6 hadoopManager

Name: pushData
Returns: Boolean
Arguments: List
Description: takes a list of events (from the event log) to push into the Hadoop filesystem, and returns a boolean with the success from the push.

Name: getData
Returns: List
Arguments: -
Description: returns the entire list of saved events.

6.7 Alpha

Name: mapReduce
Returns: Dict/Map
Arguments: List
Description: calculates the MapReduce of a given list of events.

Name: aAlgo
Returns: Net, initMark, finMark
Arguments: Dict/Map
Description: Takes a MapReduced eventlog and mines the process as a Petri Net.

6.8 backendManager

Name: getLog
Returns: List
Arguments: -
Description: to be called from outside the backend. Returns the log-list from the Hadoop system.

Name: alphaAlgo
Returns: Net, initMark, finMark
Arguments: List
Description: takes a List of events and then returns the mined Perti Net. The MR will be calculated, a call for the MR-Method is not needed.

Name: getPetri
Returns: Net, initMark, finMark
Arguments: -
Description: takes the list of events in Hadoop and calculates the Perti Net over all of them.