JOHANNES KEPLER
UNIVERSITÄT LINZ | JKU

Technisch-Naturwissenschaftliche
Fakultät

Austrian Parliament Analyzer

zur Erlangung des akademischen Grades

Bachelor of Science

im Bachelorstudium

Informatik

Eingereicht von:
Markus Hiesmair

Angefertigt am:
Institut für Telekooperation

Beurteilung:
Gabriele Anderst-Kotsis

Linz, Someday, 2015

Affidavit

I hereby declare that the following dissertation "Put your thesis title here" has been written only by the undersigned and without any assistance from third parties.

Furthermore, I confirm that no sources have been used in the preparation of this thesis other than those indicated in the thesis itself.

Linz, on October 10, 2015

Markus Hiesmair

Acknowledgment

Summary

Summary ...

Abstract

Abstract ...

Contents

Abbreviations

CAB Compute Aggregate Broadcast (A computing model in parallel computing where computation is strictly partitioned in the three phases compute, aggregate and broadcast)

IaaS Infrastructure as a Service (Cloud computing service layer)

JSF Java Server Faces (Web technology in the arena of Java enterprise)

JSP Java Server Pages (Web technology available in Java Servlet containers)

MPI Message Passing Interface (Standard for implementing parallel algorithms on shared-nothing infrastructures)

OSN Online Social Network (An usually web-based online platform where friends, and acquaintances can connect and share information)

PaaS Platform as a Service (Cloud computing service layer)

SaaS Software as a Service (Cloud computing service layer)

UML Unified Modeling Language

URL Uniform Resource Locator

VM Virtual Machine

WAR Web Application Archive

List of Figures

List of Tables

Chapter 1

Introduction

One of the most crucial requirements of a democracy is transparency. There are several ways how one can gain information about the current and past political activities in Austria. One of the best possibilities among them are the publicly available protocols of the national council sessions. In these protocols every word said in a session is written down and that makes up the corresponding protocol. Unfortunately, these protocols are very long and it is hard to gain meaning out of it, because of its plain and simple structure and the great amount of data.

To be able to analyze and visualize the activities and relations of the politicians and parties in a better way, during this thesis the protocols are being extracted, transformed, analyzed and visualized.

1.1 Research Goals

The protocols are currently available in semi-structured form - through HTML files. To be able to properly persist and analyze the data, the protocols have to be transformed into a fully structured form (e.g. Java Objects). The following elements will be extracted:

- Legislative periods and their sessions
- Politicians and their mandates
- Parliament clubs
- Discussions and speeches during the sessions

- Relations among the above listed elements

As soon as this is done, the extracted data should be persisted into a arbitrary relational database. Furthermore, some general and network analysis should be done on the data. The following list includes some of the analysis.

- Find groups of politicians (or parliament clubs) with the same attitudes.
- Analyze how homogeneous the attitudes of politicians of the same parliament club are.
- Find the politicians which take part in the most discussions.
- Find the most absent national council members.

In the final step the results should be visualized via a web application. The focus hereby lies in making the results as easy to understand as possible.

1.2 Political System in Austria

To be able to understand all the terms of the Austrian political system and to understand the general process of the law making, in this section there will be a short introduction into the field.

In Austria there is an indirect democratic system. This means that the Austrian people elect politicians (or in a more abstract way the political parties) which then represent them and try to speak for them.

1.2.1 Elections

In a national scope the most important institution is the parliament. Every five years - this is the length of a legislative period - there are federal elections. In these elections the Austrian people elect parties and depending on the results these parties get a specific number of places in the national council, called mandates. For each mandate, a party is allowed to send out a politician which can participate in national council. Altogether, there are 183 politicians in the Austrian parliament.

Chapter 2

Related Work

Chapter 3

Reference Implementation of the Distributed Processing Framework

This is a chapter that references other chapters **??**

Chapter 4

Results and Discussion

Chapter 5

Conclusions and Future Work

Appendix A

Tooling

The real world implementation of this thesis available as open source software. The code, detailed instructions, and code samples can be found at `http://www.dynamograph.net/`. The following chapter gives a brief overview of the tools in use, lists their versions, and purpose they were used for.

A.1 Java, JavaScript, Web

As highlighted in chapter 3 the prototype implementation was mainly implemented with the Java programming language. There are basically two components worth mentioning in this respect. The first component is the computing cluster which is implemented as a standalone distributed Java application that can be deployed to possibly many worker nodes in a computer cluster. The second component is the web based user interface which is implemented as a Java Enterprise web application that runs in a Servlet container. The the following paragraphs and sections describe the tools and versions used in these components.

For software project management in general and automated dependency management the infamous Apache Maven 3.1.1 toolset is used. All components of the prototype are available as a Maven configured artifacts and can be automatically built and tested on state-of-the-art software integration services such as Jenkins `http://jenkins-ci.org`. Although continuous integration is currently performed from the developers machines since the project team was rather small and most of the time consisted only of the thesis author.

Although Java 8 was already available at the time of writing this the prototype was still compiled and tested with Java 1.7.0_45-b18 since first tests with Java 8 VMs have shown that some minor changes in object serialization could cause problems with the

distributed compute cluster and the new features available in Java 8 were not used by the project anyway.

In the following list shows all Maven projects in alphabetical order and their purpose is briefly explained.

analytics: The analytics project is a helper project that contains Java and R code for general statistical analysis of datasets and is not available on production installations of the system.

census: This contains lists of firstname and lastname, and their frequency of occurance in the 1990 USA census [**?**]. The project contains a library that can be used to assign unique firstname, lastname pairs to vertices which is used to assign random names to make anonymized datasets. This process has proven to make anonymized datasets better readable.

commandline: This project contains command-line utilities that allow some control over a running temporal graph processing cluster. Mainly it can be used to properly shut down a cluster and to perform other maintenance tasks.

common: This project contains general data-structures to describe a temporal graph such as vertices and edges and also contains more general data-structures such as the implementation of a temporal map which is the basic foundation of most data components in the system.

configuration: As the name suggests this project is just a helper that allows to read configuration files from disk and provides the loaded configuration as static variables to the many other components of the system.

datasources: This project provides libraries that enable access to diverse data-sources such as static data-sets that hold temporal edge lists, relational databases, e-mail database parsers, and dynamic data-sources such as directly streaming data from Twitter.

distributed: This resembles the core of the distributed compute cluster and holds the code for all components such as the master node, the worker node, election process, graph partition table, and all the communication and multi-threading code. Implementation details of this project are described in greater detail in A.1.1.

dynamo-graph: This is an almost empty Maven parent project. It just holds a Maven project object model that can be used to build, test, and deploy all system components in one coherent build process.

frontend: Here the web based user interface is implemented. The project can be built to a standard Java EE web application contained in a WAR file and deployable to any standard compliant Servlet container as explained in more detail in A.1.3.

opencv: This project contains code that allows to reconstruct past social networks from image data. Such that user provided picture libraries can be imported and a temporal social network is extracted from them.

playground: The playground project is a potpourri of different code snippets that were used for testing. To the interested reader this project might be interesting because it shows how code can be run on top of the distributed computing framework. Especially interesting might be the parts that allow developers to run the system in so called local developer mode to debug distributed algorithms.

staticdata: This is a project that contains static data-sets used during testing and evaluating this thesis. The data-sets are mostly compiled from temporal edge lists. Each data-set available to the system also contains a configuration file that can be parsed by the frontend application and contains information like a data-set description, the color coding used and the edge types contained in the data-set.

uml: This project contains ObjectAid 1.1.6 UML class diagrams that are automatically synced with the code in the other projects. The UML diagrams are used for documentation purposes in this thesis and on the project website.

utils: Finally the utils project contains general purpose utilities that did not fit in with any other project but are used throughout many components of the project. Such utilities are a extended version of an URL resolver, parser for files encoded as comma separated values, and a tool that is capable of merging colors available as RGB color codes.

A.1.1 Distributed Java Applications

As already explained in great detail the distributed graph computing framework is implemented as a standalone Java application.

To-Do: ZooKeeper

A.1.2 Client API Methods

addEdge(String, Edge, long) addNode(String, Node) benchMarking() containsModel(String) createModel(String, Resolution) deleteModel(String) deleteVertex(String, long) deleteVertex(String, Node) executeAlgorithm(String, String) executeAlgorithm(String, String, SuperStepContext) executeAlgorithm(String, String, SuperStepContext, Timeframe) getMaximumTimeframe(String) getModelDescriptor(String) getVertex(String, long) getVertex(String, long, Timeframe) listModels() loadCode(String, byte[]) queryAlgorithmContext(long) queryAlgorithmState(long) queryExceptions(long) queryVertices(String, GraphQuery) unloadCode(String) updateVertex(String, Node) waitForAlgorithmState(long, ExecutionProfileState...) waitForCompletion(long)

A.1.3 Web-based User Interface

As explained in **??** users working with a system like the presented will most likely prefer to use web-based user interfaces. The high volume of data do be processed is just one argument for web-based access which practically allows the user to have the data completely stored in the cloud and just interface the processing from a web-browser. In the case of this project state-of-the-art web technology was used to build the user interface. The application is built as Java web application compliant with version 3.0 of the Servlet, version 2.2 of the JSP (Java Server Pages), and 2.2 of the EL (Expression Language) specifications. The visual rendering of the web pages is based on the JSF (Java Server Faces) 2.1.7 implementation provided by PrimeFaces 3.3.1 which guarantees that the user interface also complies with current standards for cross platform web-design.

For graph visualization the frontend relies on the JavaScript based graph drawing library SigmaJS 0.1 which was extended by a few simple modules which allow the user to move vertices in the graph via drag and drop, enable context information for each vertex such that computed scores for each vertex can be displayed, and the users are able to directly interact with vertices to drill deeper in a graph when necessary.

In general the web-based user interface can be built via Maven by running a build in the *frontend* project. The result is a web application archive (WAR) that can be deployed on any arbitrary Servlet 3.0 container. During the implementation and testing of this thesis mainly Apache Tomcat 7.0.29 and earlier was used and thus apache Tomcat is highly recommend for production environments also.

To-Do: add: jQuery

To-Do: add: Highcharts.js

A.2 Cloud Stacks

A.2.1 openNebula

A.2.2 openStack

A.2.3 Infrastructure Abstraction

A.2.3.1 Chef, Puppet, Dockr.io

Bibliography