Master Thesis
Computer Science

Scaling UIMA

Simon Gehring

Am Jesuitenhof 3
53117 Bonn
simon.gehring@fkie.fraunhofer.de
Matriculation Number 2553262

At the
Rheinische Friedrich-Wilhelms-Universität Bonn
in cooperation with the
Fraunhofer-Institut für Kommunikation,
Informationsverarbeitung und Ergonomie

supervised by
Prof. Dr. Heiko Röglin and [redacted]

March 1, 2018

Contents

Chapter 1

Introduction

Natural language is part of everyone's everyday life and is most commonly used to transmit information human-to-human. While most of this interaction takes place orally or written on paper, the digital revolution and the rise of social media increased the amount of digitally stored natural language tremendously. Gantz and Reinsel predicted 2012 that the amount of digital data stored globally will double about every two years until at least the year 2020 [GR12].

Many opportunities arise from this amount of digital data, specifically in the field of machine learning. In 2011, IBM's QA (Question Answering) system "Watson" famously outmatched professional players in the quiz show "Jeopardy!" [Fer12, ESI+12]. Kudesia et al. proposed 2012 an algorithm to detect so called CAUTIs[1], common hospital-acquired infections, by utilizing a NLP (Natural Language Processing) analysis on the medical records of patients [KSDG12].

Apache UIMA (Unstructured Information Management Architecture) is one of few general approaches to implement NLP solutions. With a very modular architecture, UIMA is a popular tool that can easily be applied to a majority of NLP problems. A large part of the popularity of UIMA stems from the large DKPro Core (Darmstadt Knowledge Processing Software Repository) collection of components, containing hundreds of analysis modules and precomputed language models [EdCG14], which are easily imported into existing Java projects with the build automation tool Apache Maven [DKP].

A common problem with UIMA in non-academic environments is scaling [DCR+15, ESI+12, RBJB+10]. UIMA itself provides two distinct interfaces to analyze larger collections of unstructured data, with one being UIMA-AS (UIMA Asynchronous Scaleout) and the other being the more dated and less flexible CPE (Collection Processing Engine) [FLVN09].

In this thesis, we will evaluate different means of scaling UIMA, using modern technologies like Docker, a container virtualization solution, Apache Spark, a cluster computing framework, and Apache Kafka, an information stream processing software. We will compare said implementations with the native UIMA-AS and CPE approach in terms

[1] Catheter-associated Urinary Tract Infections

of processor and memory efficiency, ease of implementation and maintainability. The evaluation will be based on a specific scenario, however it will be easily configurable by exchanging very few lines of code.

1.1 Motivation

Das wird eine kleine section. Das meiste wird aus Introduction rausgenommen. Und der Text direkt unter Introduction ist als Einleitung in das Kapitel zu verstehen und daher nur 1-3 Absätze groß. Einleitungception.

1.2 Basics

In this chapter, we will cover the basics for the necessary technologies used throughout the evaluation. All of these are concrete implementations of more general concepts and may be exchanged for similar products. However, the following products were chosen, mainly because they are Open Source[1,2,3,4,5] but also because of their popularity and relevance in the industry.

1.2.1 UIMA

UIMA is a data mining and NLP framework, created in 2005 by IBM [FL04] and maintained by Apache since 2006 [Apac]. It is available in Java and C++ and contains various scale-out options. Strictly speaking, one has to differentiate between the UIMA specification and Apache UIMA, an open source implementation of said specification [FLVN09]. Since both terms are often used interchangeably, this thesis will also not differentiate between Apache UIMA and UIMA. Unless specified else, we will always reference the Apache UIMA implementation.

Alles ein wenig wirr. Braucht rewrite. Weiß aber nicht genau wie tief in die Materie ich hier gehen muss.

The UIMA specification describes NLP applications as a collection of (mostly) independent components. Such a component is called an AE (Analysis Engine) and enriches a given document by inferred information. To modularize said components, UIMA provides the notion of a CAS (Common Analysis System). A CAS is an object containing the SofA (Subject of Analysis), the analysis results and the used type system.

1.2.2 Docker

Source?

Docker is a software to create and run applications in virtual environments, without the need to setup a complete virtual operating system for each application. Based on Cgroups and namespaces, Docker runs applications in a containerized environment. To create a containerized application, a special markup file, called a Dockerfile has to be written by the programmer. The Dockerfile defines exactly how the applications' environment needs to look like. Listing 1.1 shows how such a Dockerfile typically looks

[1] `https://svn.apache.org/viewvc/uima/`, last accessed on 2018-02-27.
[2] `https://github.com/docker`, last accessed on 2018-02-27.
[3] `https://github.com/apache/hadoop`, last accessed on 2018-02-27.
[4] `https://github.com/apache/spark`, last accessed on 2018-02-27.
[5] `https://github.com/apache/kafka`, last accessed on 2018-02-27.

like. Based on an Ubuntu image, it installs Java via apt-get. Afterwards it copies the file *application.jar* into the root of the image and tells Docker to execute it via `java -jar /app.jar -Xmx4G -Xms4G`. Having such a Dockerfile ensures reproducibility of the constructed images. After building said Dockerfile, an image is created. This image can be serialized into a file but is usually uploaded to Docker repositories, called registries.

source für Registries? Öffentliches Docker-Registry?

```
1 FROM  ubuntu
2
3 RUN   apt-get update
4 RUN   apt-get install openjdk-9-jre
5
6 COPY  ./application.jar /application.jar
7
8 ENTRYPOINT ["java", "-jar /app.jar", "-Xmx4G", "-Xms4G"]
```

Listing 1.1: A sample Dockerfile used for creating a simple Ubuntu based image to start a Java application.

If published through a registry, Docker automatically downloads images on demand. Since the application as well as the whole environment necessary to run it lie within the image, it can be executed. Docker creates a container, a virtual environment based on the given image, and runs the application within. Containers are usually not aware of each other, thus the same application can be started multiple times, just by starting more containers. This modular view on containers will play a fundamental role on scaling.

SOUUUURCE!!

Zweimal within? Thesaurus lässt grüßen.

An den Haaren herbeigezogen.

1.2.3 Hadoop

1.2.4 Spark

1.2.5 Kafka

1.3 Problem

1.3.1 Scaling UIMA

UIMA-CPM

UIMA-AS

UIMA-DUCC

1.3.2 Implementation Requirements

1.4 Related Work

1.4.1 Watson

1.4.2 Something else that warrants another subsection

1.4.3 GATE?

1.5 Outline

Chapter 2

Implementation

2.1 Concrete Application

2.2 Documents -> Kafka

2.3 Kafka -> Spark

2.4 Spark -> UIMA

2.5 UIMA -> Java

2.6 Kafka -> Output

2.7 Bottlenecks

Chapter 3

Evaluation

3.1 Computation Speed

3.2 Memory Usage

3.3 Extensibility

3.4 Maintainability

Chapter 4

Summary

4.1 The Judgement

Chapter 5

Future Work

Glossary

Analysis Engine

An Analysis Engine is a component of an UIMA NLP pipeline. As such it analyses given documents and enriches it with inferred information. An Analysis Engine may contain other Analysis Engines [Apac]. 2

Apache Kafka

Gestohlener Text, muss noch paraphrasiert werden.

Apache Kafka is an open-source stream processing software platform developed by the Apache Software Foundation written in Scala and Java. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. 1

Apache Spark

Gestohlener Text, muss noch paraphrasiert werden.

Apache Spark is an open-source cluster-computing framework. Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. 1

Collection Processing Engine

Collection Processing Engines (CPE) are the first generation of UIMA native scaling solutions. A CPE contains a collection reader, which knows how to read the underlying collection, and CAS Consumers for the final analysis result extraction [Apac]. 1

Common Analysis System

The Common Analysis System is a type of object in the UIMA framework. It contains the subject of analysis, the analysis result and a corresponding type system [Apac]. 2

Darmstadt Knowledge Processing Software Repository

A collection of UIMA components for natural language processing. This includes analysis engines, language models and custom type systems [DKP, EdCG14]. 1

Docker

Docker is a virtualization solution based on containers. By using containers instead of fully fledged virtual machines Docker tries to reduce the system overhead per running application [doc15]. i, 1, 2, 3

Natural Language Processing

Natural-language processing (NLP) is the discipline of collecting and analysing natural language. This includes for example speech recognition, natural language understanding and generation [Lid01]. 1

Question Answering

Being a subfield of NLP, Question Answering (QA) is about extracting and understanding questions from natural language and answering them accordingly [JM14]. 1

Subject of Analysis

The Subject of Analysis is the document that gets analyzed by a given UIMA application. It is contained in its corresponding CAS [Apac]. 2

UIMA Asynchronous Scaleout

UIMA-AS is the second generation of UIMA native scaling solutions. It is based on a shared queue based service architecture [Apab] 1

Unstructured Information Management Architecture

UIMA is a general purpose framework to extract information from unstructured data [Apaa, FLVN09]. Although any data format is supported, natural language texts are the most common one. 1

Bibliography

[Apaa] The Apache Software Foundation. Apache uima – apache uima. https://uima.apache.org/. Accessed: 2018-02-26.

[Apab] The Apache Software Foundation. Getting started: Apache uima asynchronous scaleout. https://uima.apache.org/doc-uimaas-what.html. Accessed: 2018-02-26.

[Apac] The Apache Software Foundation. Uima tutorial and developers' guides. https://uima.apache.org/d/uimaj-2.4.0/tutorials_and_users_guides.html. Accessed: 2018-02-26.

[DCR+15] Guy Divita, M Carter, A Redd, Q Zeng, K Gupta, B Trautner, M Samore, and A Gundlapalli. Scaling-up nlp pipelines to process large corpora of clinical notes. *Methods of information in medicine*, 54(06):548–552, 2015.

[DKP] The DKPro Core Team. Dkpro core™ user guide. https://zoidberg.ukp.informatik.tu-darmstadt.de/jenkins/job/DKProCoreDocumentation(GitHub)/de.tudarmstadt.ukp.dkpro.core$de.tudarmstadt.ukp.dkpro.core.doc-asl/doclinks/6/user-guide.html. Accessed: 2018-02-26.

[doc15] What is docker? https://www.docker.com/what-docker, 2015. Accessed: 2018-02-26.

[EdCG14] Richard Eckart de Castilho and Iryna Gurevych. A broad-coverage collection of portable nlp components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland, August 2014. Association for Computational Linguistics and Dublin City University.

[ESI+12] Edward A Epstein, Marshall I Schor, BS Iyer, Adam Lally, Eric W Brown, and Jaroslaw Cwiklik. Making watson fast. *IBM Journal of Research and Development*, 56(3.4):15–1, 2012.

[Fer12] David A Ferrucci. Introduction to "this is watson". *IBM Journal of Research and Development*, 56(3.4):1–1, 2012.

[FL04] David Ferrucci and Adam Lally. Uima: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, 2004.

[FLVN09] David Ferrucci, Adam Lally, Karin Verspoor, and Eric Nyberg. Unstructured information management architecture (UIMA) version 1.0. OASIS Standard, mar 2009.

[GR12] John Gantz and David Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, 2007(2012):1–16, 2012.

[JM14] Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London:, 2014.

[KSDG12] Valmeek Kudesia, Judith Strymish, Leonard D'Avolio, and Kalpana Gupta. Natural language processing to identify foley catheter–days. *Infection control and hospital epidemiology*, 33(12):1270–1272, 2012.

[Lid01] Elizabeth D Liddy. Natural language processing. 2001.

[RBJB+10] Cartic Ramakrishnan, William A Baumgartner Jr, Judith A Blake, Gully APC Burns, K Bretonnel Cohen, Harold Drabkin, Janan Eppig, Eduard Hovy, Chun-Nan Hsu, Lawrence E Hunter, et al. Building the scientific knowledge mine (sciknowmine): a community-driven framework for text mining tools in direct service to biocuration. *Language Resources and Evaluation*, page 33, 2010.

Eidesstattliche Erklärung

Hiermit versichere ich, Simon Gehring, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken und Quellen, einschließlich Quellen aus dem Internet, entnommen sind, habe ich in jedem Fall unter Angabe der Quelle deutlich als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen, Karten und Abbildungen.

Unterschrift: _____

Simon Gehring, Student
Universität Bonn