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 Dr. Timm Heuss

March 5, 2018

Contents

Abstract

Natural language 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 with precomputed language models on the medical records of patients [KSDG12].

Natural language unfortunately tends to be unstructured and hardly machine readable. Even a seemingly easy task, like separating a sentence into words is still an ongoing research topic [PT18]. 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 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.

The following decisions are still to be made:

- On what level of abstraction will be parallelized? (Inter-Annotator, Inter-Pipeline, Inter-JVM, ...)

[1]Catheter-associated Urinary Tract Infections

- What methods of parallelization will be used? (Spark? Native Java Threads? UIMA-AS? Services?)
- What application (and implied data set) is going to be parallelized?
- What metrics are we trying to optimize?
  - CPU Workload (Does the system use all the available processing power?)
  - RAM Usage (How much RAM does the system use? Does it swap?)
  - Throughput (How many documents per second are processed?)
  - Maintainability (Are infrastructure changes or code changes easy to implement?)
  - Extensibility (How easy is it to add more processing power?)
  - Implementability (How simple is the implementation?)

Chapter 1

Introduction

1.1 Motivation

1.2 Problem

1.2.1 Scaling UIMA

UIMA-CPM

UIMA-AS

UIMA-DUCC

1.2.2 Implementation Requirements

1.3 Related Work

1.3.1 Watson

1.3.2 v3NLP

1.3.3 GATE?

1.4 Outline

Chapter 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]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.

Chapter 3

Implementation

3.1 Concrete Application

3.2 Documents $\rightarrow$ Kafka

3.3 Kafka $\rightarrow$ Spark

3.4 Spark $\rightarrow$ UIMA

3.5 UIMA $\rightarrow$ Java

3.6 Kafka $\rightarrow$ Output

3.7 Bottlenecks

Chapter 4

Evaluation

4.1 Computation Speed

4.2 Memory Usage

4.3 Extensibility

4.4 Maintainability

Chapter 5

Summary

5.1 The Judgement

Chapter 6

Future Work

Glossary

Apache Kafka

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

Gestohlener Text, muss noch paraphrasiert werden.

Apache Spark

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

Gestohlener Text, muss noch paraphrasiert werden.

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

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]. 1

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

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.

[DG08] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[DKP] The DKPro Core Team. Dkpro core$^{\text{TM}}$ 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.

[PT18] Irina Pak and Phoey Lee Teh. Text segmentation techniques: A critical review. In *Innovative Computing, Optimization and Its Applications*, pages 167–181. Springer, 2018.

[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.

[Wik17] Wikipedia. Mapreduce — wikipedia, die freie enzyklopädie, 2017. [Online; Stand 2. März 2018].

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