# Using Apache Spark on Hadoop Clusters as Backend for WebLicht Processing Pipelines

**Soheila Sahami**
sahami@informatik.uni-leipzig.de

**Thomas Eckart**
teckart@informatik.uni-leipzig.de

**Gerhard Heyer**
heyer@informatik.uni-leipzig.de

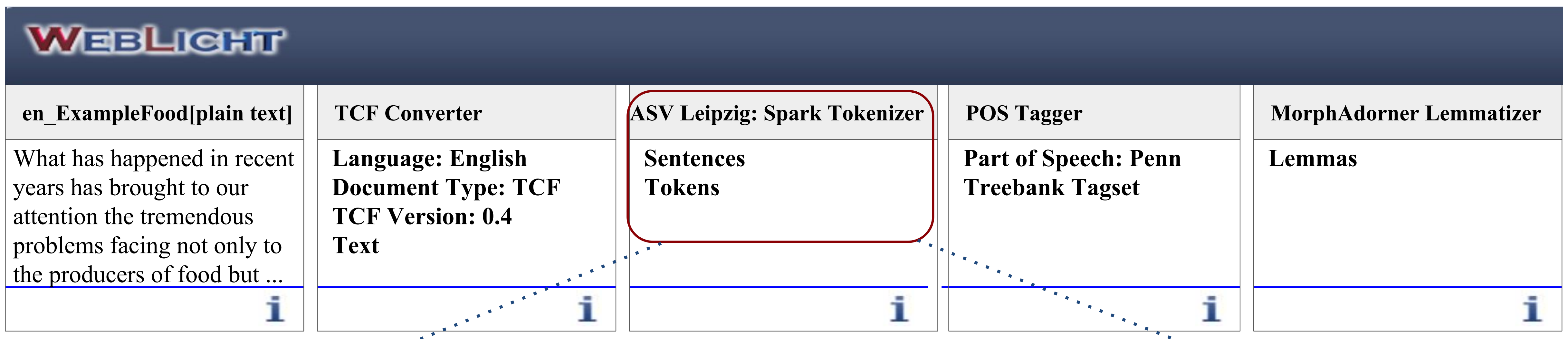
**Natural Language Processing Group, University of Leipzig**

## 1. Problem Statement

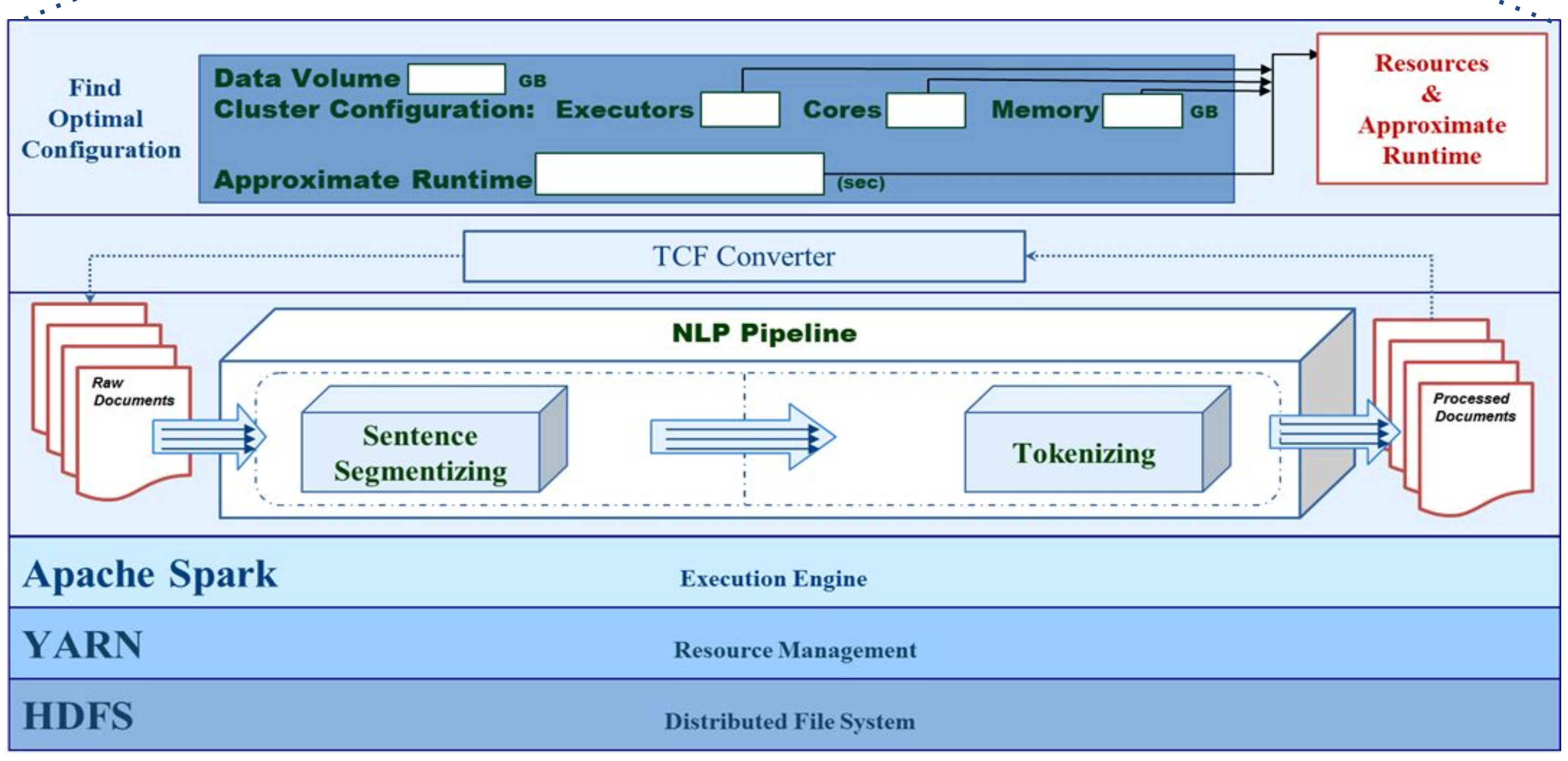
Service oriented architecture (SOA)-based platforms - like CLARIN's WebLicht - are a solution for modern annotation tools and pipelines that support automatic text annotation and processing to simplify their active use and to relieve users from complex configuration tasks.

However, in many cases the current state of participating endpoints does not allow processing of big data or the execution of many user tasks in parallel.

Besides, required hardware resources and runtimes to process big volume input materials are still major challenges.



| en_ExampleFood[plain text] | TCF Converter | ASV Leipzig: Spark Tokenizer | POS Tagger | MorphAdorner Lemmatizer |
|---|---|---|---|---|
| What has happened in recent years has brought to our attention the tremendous problems facing not only to the producers of food but ... | Language: English<br>Document Type: TCF<br>TCF Version: 0.4<br>Text | Sentences<br>Tokens | Part of Speech: Penn Treebank Tagset | Lemmas |



## 2. Technical Approach

Apache Hadoop provides a framework for the distributed processing of large data sets.

Hadoop Distributed File System (HDFS) is a distributed storage solution for processing large data sets with eminently fault-tolerant and high-throughput access to application data.

Apache Spark, a cluster computing platform which can be used as execution engine to process huge data sets, uses a multi-threaded model and In-Memory Databases (IMDB) to improve processing times and fault tolerance.

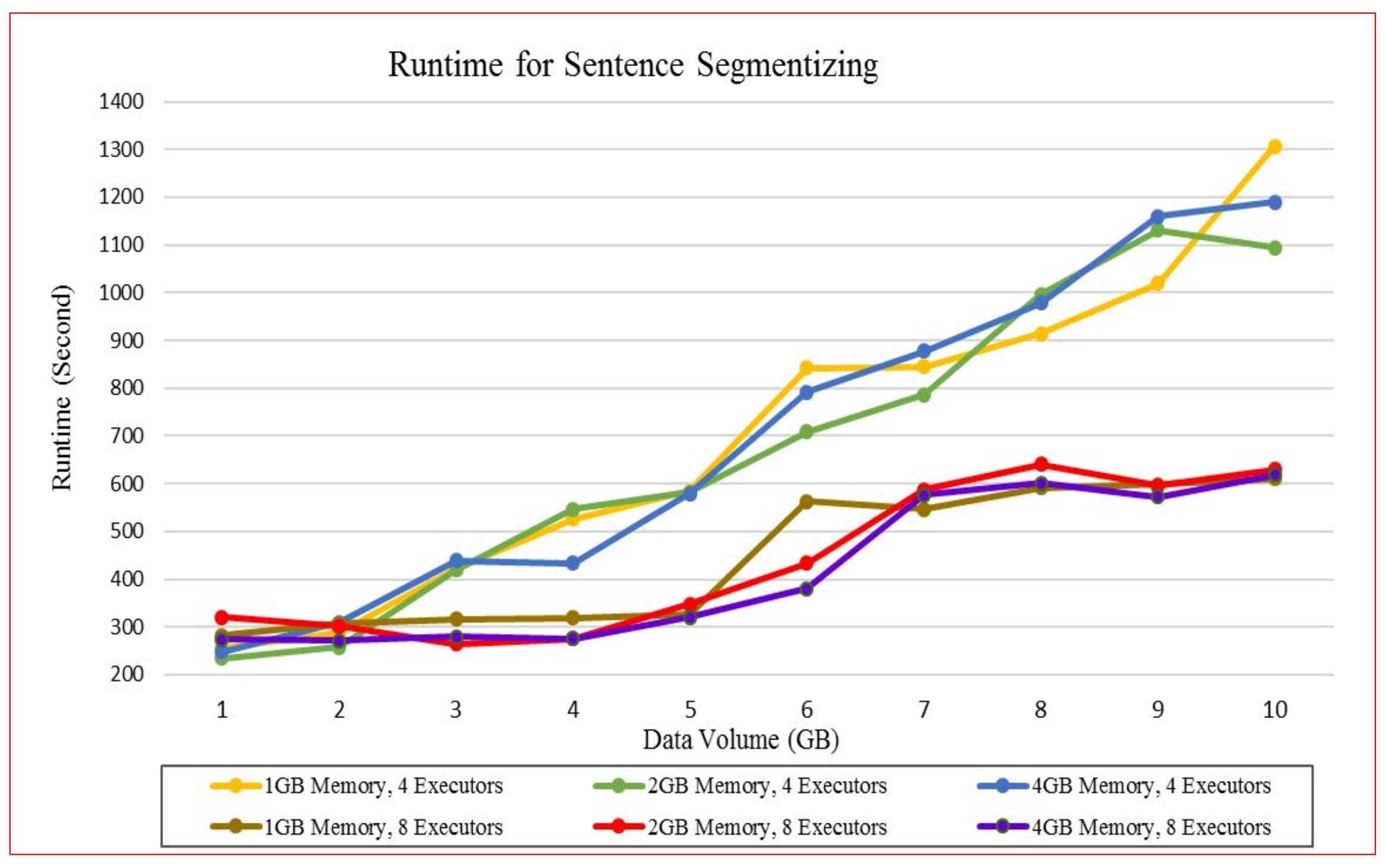
Utilization of a Hadoop/Spark-based system fits into the service-oriented WebLicht architecture and can provide enhanced processing capabilities to the infrastructure.

## 3. Implementation and Results

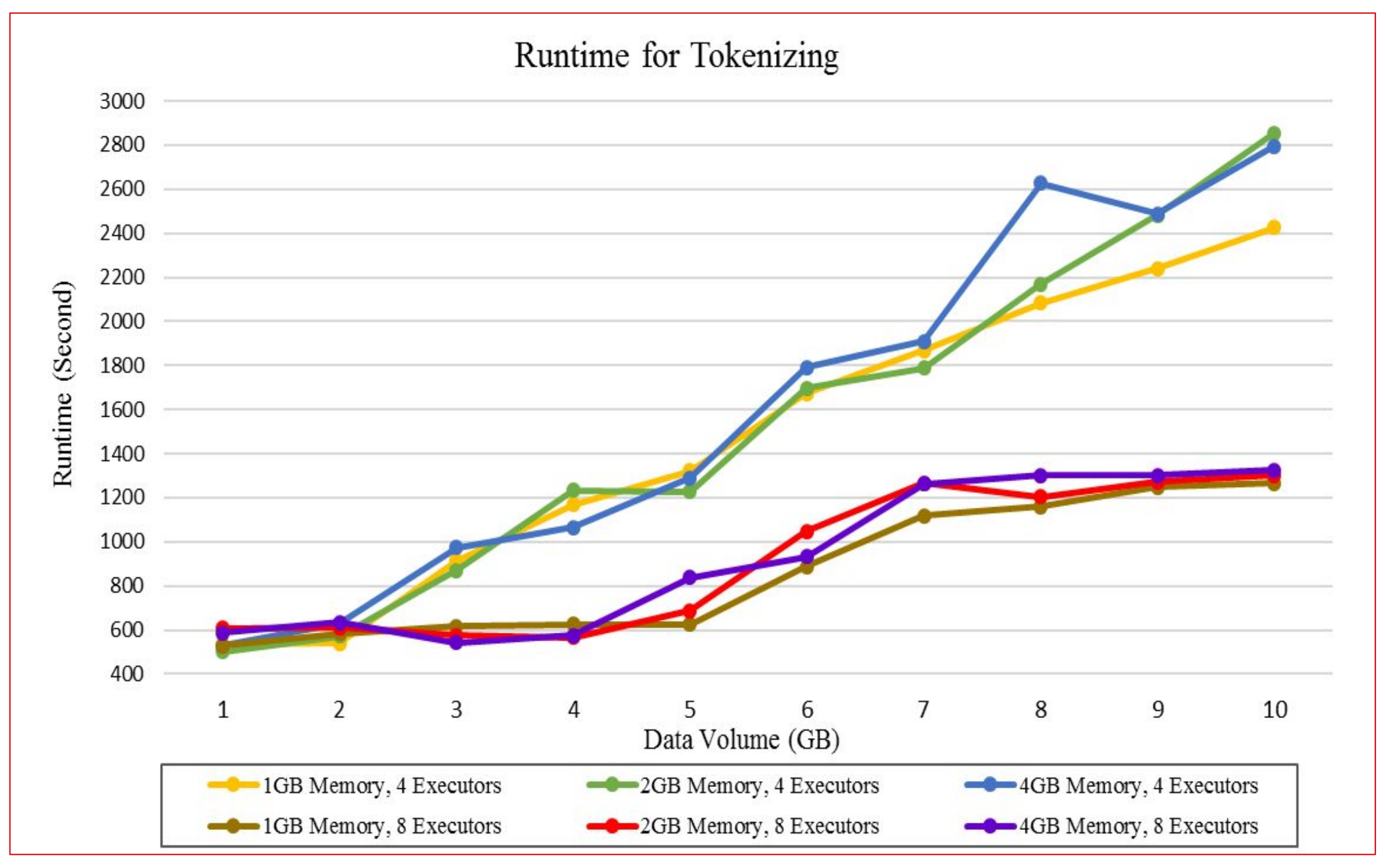
In the current working prototype, a variety of typical NLP tools - including sentence segmentation, pattern-based text cleaning, tokenizing, and language identification - are implemented. During the execution, input text files are loaded into Spark data sets (RDD) and distributed over the allocated cluster hardware to be processed in parallel. The specific configuration is determined automatically based on empirical values taken from previous runs and takes the current workload of the underlying cluster into account.

For every job, the hardware configuration can be set dynamically, considering volume and type of input data as well as the selected processing pipeline which may consist of a single or even multiple tools.

For evaluation of the established solution, benchmarks were executed to show the impact of parallelization on every task. In these experiments, we executed each tool with different resources - from 1 to 16 gigabytes of RAM and 1 to 16 executors - on various data volumes from 1 to 100 gigabytes. The following diagrams show the run time for different configuration and varying size of input documents.



Sentence Segmentizing 1 to 10 GB text data
using 4 or 8 executors and 1, 2 or 4 GB RAM



Tokenizing 1 to 10 GB text data
using 4 or 8 executors and 1, 2 or 4 GB RAM

## 4. Outcomes

- ❑ Support of processing large amount of text material (big data) without loosing the benefits of a service-oriented architecture.
- ❑ Efficient use of parallelization, including the parallel processing of large document collections and the support of large user groups.
- ❑ Open accounting of used resources (ranging from used hardware resources to financial costs) for enhancing user acceptance of services and workflows by making hidden costs more transparent.
- ❑ Dynamic configuration of resources based on the volume and type of the input texts.
- ❑ Using IMDB property of Apache Spark to keep middle results in memory and no need to transfer big data over the network.
- ❑ Central storage solutions (like private workspace environments) may be beneficial for processing large data sets in a SOA.

**References:**

1. [Apache Hadoop2018] Apache Hadoop. 2018. Apache Hadoop Documentation. Online. Date Accessed:11 Apr 2018. URL http://hadoop.apache.org/.
2. [Enslow1978] Philip H. Enslow. 1978. What is a "distributed" data processing system? Computer,11(1):13–21.
3. [Gate Cloud2018] Gate Cloud. 2018. GATE Cloud: Text Analytics in the Cloud. Online. Date Accessed: 11 Apr 2018. URL https://cloud.gate.ac.uk/.
4. [Hinrichs et al.2010] Erhard W. Hinrichs, Marie Hinrichs, and Thomas Zastrow. 2010. WebLicht: WebBased LRT Services for German. In Proceedings of the ACL 2010 System Demonstrations, pages 25–29.
5. [Karau et al.2015] Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia. 2015. Learning spark: lightning-fast big data analysis. O'Reilly Media, Inc.
6. [Lars-Peter Meyer2018] Lars-Peter Meyer. 2018. The Galaxy Cluster. Online. Date Accessed: 12 Apr 2018. URL https://www.scads.de/de/aktuelles/blog/264-big-data-cluster-in-shared-nothingarchitecture-in-leipzig.
7. [PRACE2018] PRACE. 2018. PRACE Research Infrastructure. Online. Date Accessed: 27 Apr 2018.URL http://www.prace-ri-eu.
8. [Rahm and Nagel2014] Erhard Rahm and Wolfgang E. Nagel. 2014. ScaDS Dresden/Leipzig: Ein serviceorientiertes Kompetenzzentrum für Big Data. In E. Plödereder, L. Grunske, E. Schneider, and D. Ull, editors, Informatik 2014, pages 717–717, Bonn. Gesellschaft für Informatik e.V.
9. [TCF2018] 2018. The TCF Format. Online. Data Accessed: 27 Apr 2018. URL https://weblicht.sfs.unituebingen.de/weblichtwiki/index.php/The_TCF_Format.

Printed at the central computing center of University of Leipzig