Orchestrating the Natural Language Processing Software in the Cloud Computing Environment

Dmitry Ustalov, Mikhail Goldshtein
Institute of Mathematics and Mechanics UrB RAS
Russia
{dau, mlg}@imm.uran.ru

Journal of Digital Information Management

ABSTRACT: *The most of natural language processing problems are data-intensive. An important step in the distributed orchestration of natural language processing software is a rational choice of the specific middleware. The middleware should solve the presented problem with minimal deployment, support and usage costs. It is necessary to run and use that software in the distributed cloud computing environment to achieve such advantages such as consolidation, isolation, and efficient use of the existent infrastructure. It is often impossible to modify the existent natural language processing software to integrate it into the cloud computing environment because of licensing or organizational issues. This paper studies various popular distributed data processing tools and evaluates the selected natural language processing tools on a relatively large document collection in distributed way using the Gearman framework. The document collection is a 10'000 sentences from the Russian news subcorpus of the Leipzig corpora. The benchmarks are presented and discussed.*

Categories and Subject Descriptors:
I.2.7 [Natural Language Processing]; **C.2.4 [Distributed Systems]**

General Terms:
Natural Language Processing, Distributed Systems

Keywords: Cloud Computing, Data-intensive Computing, Distributed Computing, Natural Language Processing, Service Orchestration

Received: 11 July 2013, Revised 19 August 2013, Accepted 24 August 2103

1. Introduction

The most of natural language processing (NLP) problems are data-intensive problems. For instance, such tasks as named entity recognition, automatic summarization, machine translation, etc.

In view of wide popularity of the cloud computing technology and large development of the cloud-based natural language processing systems (i.e. GATE Cloud [3]), the problem of creating the distributed text processing systems becomes more actual because of high requirements to natural language text processing frameworks that works in cloud.

Nowadays there are many production-grade NLP software. This software is mostly standalone and not distributed by design. However, it is necessary to run and use that software in the distributed cloud computing environment to achieve such advantages such as consolidation, isolation, and efficient use of the existent infrastructure.

An important step in the distributed orchestration of natural language processing software is a rational choice of the specific middleware. The middleware should solve the presented problem with minimal deployment, support and usage costs.

2. Distributed Data Processing Tools

The first study considers and compares the popular distributed data processing orchestration middleware such as Apache Hadoop [1], Apache Thrift [2], Gearman [4], and Storm [9].

The comparison used the following criteria:

• The modification necessity of the NLP software, the count of points of failure (POFs),

• The activity state of ecosystem and user community of

• The middleware, the ease of middleware administration,

• The ease of middleware using to an end user.

Table 1 presents the main results of this study.

Criterion	Hadoop	Thrift	Gearman	Storm
Modification	No	Yes	No	Yes
POFs	1	1	1	1
Ecosystem	High	Medium	Medium	Medium
Ease of A.	Medium	High	High	Medium
Ease of U.	Medium	Medium	High	Medium

Table 1. The distributed data processing tools comparison

It is often impossible to modify the existent NLP software to integrate it into the cloud computing environment because of licensing or organizational issues. Such middleware as Gearman and Hadoop Streaming allow to evade these issues and to provide the necessary natural language processing services to the end users.

Every middleware under this study has a single point of failure (SPOF). For example, a NameNode for Hadoop, a Thrift-server for Thrift, a `gearmand` process for Gearman and a Nimbus node for Storm. It is often possible to finish the correctly started job when the orchestration middleware is unavailable.

Apache Hadoop has the most developed ecosystem and community, but supporting and using the Hadoop infrastructure require an advanced level of both administrators and users experience.

The formulation of several NLP problems is complicated in the terms of the MapReduce paradigm that Hadoop uses. Thus, Apache Hadoop features are becoming unengaged in a number of the practical natural language processing tasks. In this case, the usage of Gearman became easier because of its simple three-level "*clients* ↔ *job server* ↔ *workers*" architecture.

3. Natural Language Processing Tools

Various NLP tools are designed to solve different natural language processing problems. Some of these problems require huge dictionaries and databases to be available.

The second study evaluates the scalability of three NLP tools for the Russian language processing that are popular at the NLPub linguistic knowledge base [7]. This study is considering the following properties of the software to be orchestrated:

• *Initialization time* is an amount of time needed to initialize the analyzer,

• *Processing time* is an amount of time that analyzer requires to process one piece of the input data.

Greeb is a text tokenization and segmentation tool that is written in Ruby [5]. Despite of the dynamic and interpreting nature of the Ruby programming language, its initialization time is insignificant and the processing time is insignifi-cant, too.

TreeTagger is a part-of-speech tagger and a lemmatization tool that is written in C++ [10]. The tagger requires significant time to initialize and significant time to process the given text.

Link Grammar Parser is a natural language parser that is written in C [6]. The parser requires significant time to initialize its dictionaries, but performance time is insignificant.

This study uses the latest versions of the described software: Greeb 0.2.2, TreeTagger 3.2, Link Grammar Parser 4.8.0.

4. Evaluation

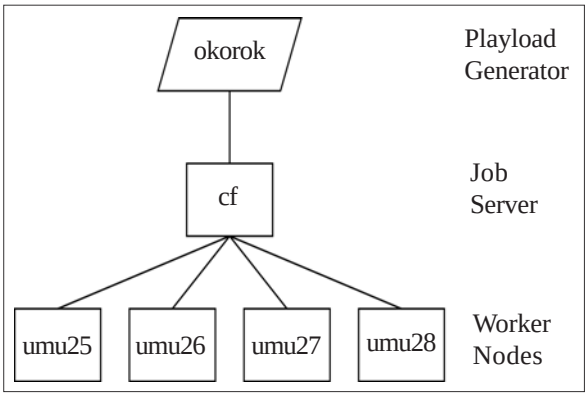

Figure 1. The experiment configuration

The comparison was performed to evaluate the performance of the described natural language processing software. Extracted sentences of the Russian 2010-*news*-10*K* subcorpus of the Leipzig corpora [8] was used as an evaluation payload. The evaluation scheme is presented at Figure 1.

The experiment has been performed in IMM UrB RAS computation centre under the following conditions for every presented natural language processing tool [11]:

1. `cf` job server node with one worker node `umu25`;

2. `cf` job server node with two worker nodes: `umu25` and `umu26`;

3. `cf` job server node with three worker nodes: `umu25`, `umu26`, and `umu26`;

4. `cf` job server node with four worker nodes: `umu25`,

umu26, umu27, and umu28.

In the experiment, the NLP tools were run under the Gearman framework that provides an orchestration of distributed work of these analyzers. Gearman job server was deployed at cf node and performs the load balancing between worker nodes: umu25, umu26, umu27, and umu28. The job server accepts requests and communicates with worker nodes through its own binary Gearman protocol.

Job server is deployed at the cf node, and payload generator is at the okorok node. Both of these nodes are the equiv-alently configured virtual machines with CentOS 6.4 (x86_64) operating system, and installed at the VMware ESXi 5.0 hypervisor. The hypervisor is running at the HP ProLiant DL165 G7 6172 server of the following configuration:

- Two 12-core AMD Opteron™ 6172 @ 2.1 GHz processors (there are only 3 cores available to virtual machines);
- 16 GB of RAM (each virtual machine has its dedicated 4 GB);
- 10 GB of virtual HDD is available to every virtual machine;
- HP NC362i Integrated Dual Port Gigabit Server Adapter.

The worker nodes umu25, umu26, umu27, and umu28 are working under CentOS 6.4 (x86_64) operating system that had been installed at four equivalent bare metal Fujitsu-Siemens Computers PRIMERGY RX330 S1 servers of the following configuration:

- Two 2-core AMD Opteron™ 2218 @ 2.6 GHz processors;
- 8 GB of RAM;
- Seagate Barracuda ES 250GB Serial ATA II 7200RPM 16MB hard disk drive;
- Broadcom BCM5715 Gigabit Ethernet network adapter.

Every described node is connected by dual-duplex Gigabit Ethernet technology. Each worker node is running four instances of the Gearman worker process that execute NLP software under evaluation. The payload generator sends one sentence per job and maintaining 16 parallel jobs workload.

	Active Worker Nodes			
	1	2	3	4
Greeb	4m 58s	3m 49s	3m 48s	3m 57s
TreeTagger	26m 52s	13m 27s	8m 58s	8m 22s
Link Grammar Parser	180h 02m 18s	83h 23m 07s	42h 31m 35s	19h 59m 14s

Table 2. Evaluation results

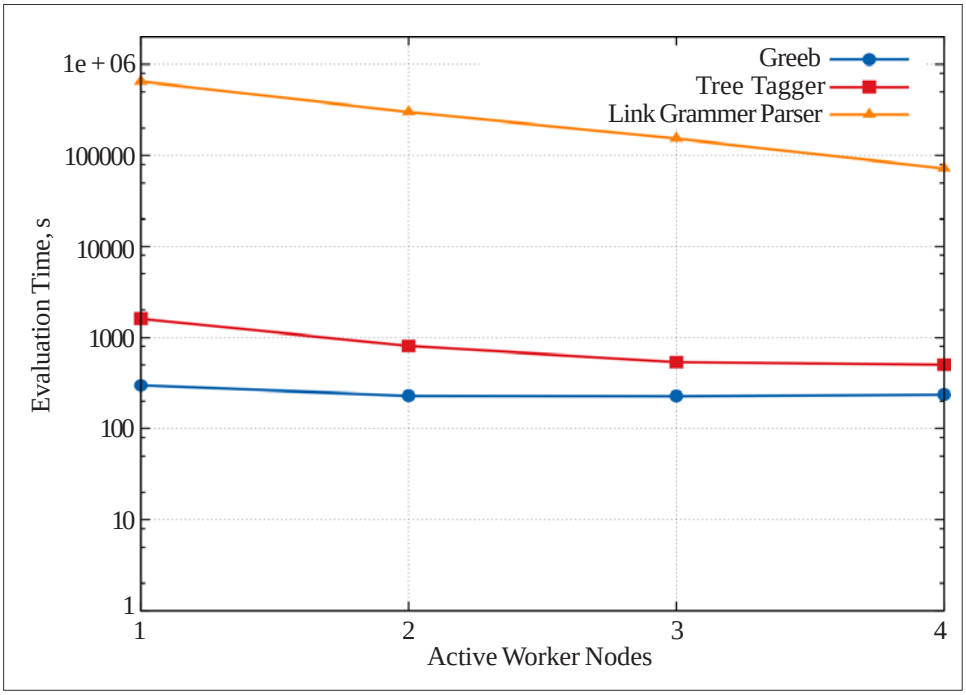

Figure 2. Evaluation of the NLP software

The results are presented at Table 2 and are depicted at Figure 2.

5. Discussion

Accoriding to Table 2, it's clear to highlight three different patterns, which are correspondent to the specific software behavior.

The NLP software with both *insignificant initialization and processing time* will not achieve huge performance boost from the distributed environment. For instance, the Greeb plot at Figure 2 demostrates the significant presence of intercommunication overhead. The minimal total processing time is observed at three worker nodes instead of four worker nodes. This drawback could be overcome by transferring data to be processed closer to the processing nodes, like Apache Hadoop does.

From the other side, it is easy to achieve significant performance boost for NLP software with both *significant ini-tialization and processing time* in the distributed environment, but this boost has narrow software-dependent limits. Specifically for *TreeTagger* the boost stops being linear starting with two active worker nodes.

The benchmark is performed using unmodified (sometimes called "*vanilla*") NLP software. In certain circumstances it is necessary to modify the software to be effective in the distributed environment. For example, the performance of *Link Grammar Parser* is unacceptable because of very significant initialization time and insignificant processing time. In such cases it would be useful to develop wrappers that could initialize once and interact with Gearman directly using its binary protocol.

6. Conslusion

Two studies were presented:

• The comparison study of distributed data processing middleware which focuses on the ease of administration and the ease of using to an end user,

• The evaluation study of the Gearman framework on a relatively large document collection.

The studies allow to make the following consequences:

• The Gearman framework performs well on software that have both significant initialization and processing time with minimal intercommunication overhead;

• The highlighted patterns allow to analyze the cloud-based NLP software and optimize such software using the discussed patterns.

References

[1] Apache™ Hadoop ®. http://hadoop.apache.org/.

[2] Apache™ Thrift ®. http://thrift.apache.org/.

[3] GATE Cloud — a New Way to Mine the Web. http://gatecloud.net/.

[4] Gearman Job Server. http://gearman.org/.

[5] greeb | RubyGems.org. https://rubygems.org/gems/greeb.

[6] Link Grammar. http://www.abisource.com/projects/link-grammar/.

[7] NLPub — Russian Computational Linguistics Wiki. http://nlpub.ru/.

[8] Uwe Quasthoff, Matthias Richter, Christian Biemann. (2006). Corpus Portal for Search in Monolingual Corpora. *In*: Proceedings of the fifth international conference on Language Resources and Evaluation, LREC 2006, p. 1799–1802.

[9] Storm. http://storm-project.net/.

[10] TreeTagger. http://www.cis.uni-muenchen.de/˜schmid/tools/TreeTagger/.

[11] Dmitry Ustalov, Mikhail Goldshtein. (2012). A distributed dictionary-based morphological analysis framework for Russian language processing. Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta, Seriya *Matematich-eskoe modelirovanie i programmirovanie*, 13 (27) 119–127.