Instituto de Engenharia de Sistemas e Computadores
Investigação e Desenvolvimento em Lisboa
Laboratório de Sistemas de Língua Falada

NLE-GRID T5 – Performance Experiments

Natural Language Engineering on a Computational Grid POSC/PLP/60663/2004

— INESC-ID Lisboa Tech. Rep. 35/2008 —

Tiago Luís, David Martins de Matos, Ricardo Ribeiro
L^{2}F – Spoken Language Systems Laboratory
INESC ID Lisboa, Rua Alves Redol 9, 1000-029 Lisboa, Portugal
{tmcl,david,rdmr}@l2f.inesc-id.pt

This report describes the application used to test the G8 platform and presents results obtained from running it on a Hadoop backend.

This revision: January 31, 2008

NLE-GRID T5: Performance Experiments Natural Language Engineering on a Computational Grid POSC/PLP/60663/2004

Tiago Luís, David Martins de Matos, Ricardo Daniel Ribeiro

L^{2}F – Spoken Language Systems Laboratory
INESC ID Lisboa, Rua Alves Redol 9, 1000-029 Lisboa, Portugal
{tmcl,david,rdmr}@l2f.inesc-id.pt

Abstract. This report describes the application used to test the G8 platform and presents results obtained from running it on a Hadoop backend.

This report documents the deployment of an application on the NLE-GRID infrastructure. The applications were built upon the Hadoop backend and deployed according to the architectural specifications [2] (figure 1). Note that by integrating them in this way, they automatically become services available in the higher levels.

One of the applications was later incorporated in an end-user application. This application, a search engine (and the corresponding web interface) providing post-processed answers for improved results, relies on statistical processing of the obtained results. The post-processing operations are computationally intensive for a single machine to run and are not at all suitable for a web application (in some cases, processing time is in excess of 15 minutes on a single machine). The application, thus, tests two aspects: the infrastructure itself and the access capabilities, on the one hand, and our capability to provide a high performance grid service through an implementation of the Hadoop backend.

The Hadoop backend is integrated in the architecture as shown in figure 1 [2].

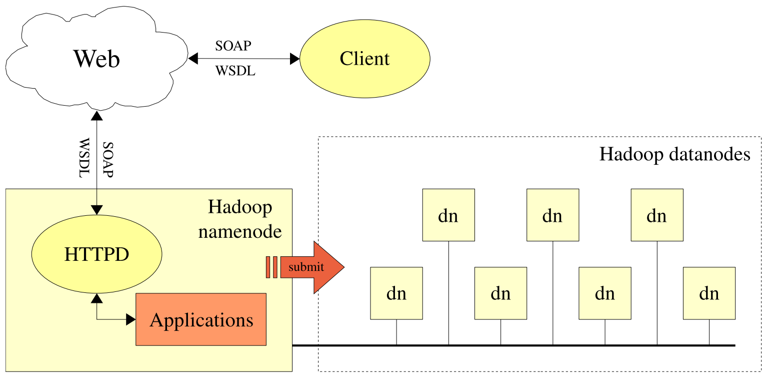

Fig. 1. Block diagram of the Hadoop service connector.

The rest of the report starts with the description of the n-gram counting application as implemented over Hadoop (§1); then, in §2, the experiments run to validate the system are presented. Finally, remarks on the performance and future applications are discussed.

Other tests to the services interface and to the web portal were also conducted. However, those tests were mainly to validate choices and particular implementation aspects and are not discussed in this report. The applications described here correspond to data-heavy processing applications and are intended to demonstrate the benefits of our approach.

1 Applications

In this section we describe two of the applications used to validate our design approach. Both of these applications explore the distributed computation available in our network setup. The reason is that potentially interested parties will have greater need of more powerful grid nodes, as opposed to the ones providing specific functionality, although these are also important. The difference is in how difficult is to achieve the same functionality (in each case) without resorting to a grid environment.

1.1 N-gram counting

The application is an n-gram counter with optional filtering. The basic operation consists of splitting the input into lines and phrases and then counting the sequences comprised between two bounds. In addition, filters are applied to the proposed n-grams: the main filter types are stop list filters, structural filters, and punctuation filters.

An effort was made to keep the application language-independent: this means that the the algorithms for language processing used, for instance, in filters, must also be language independent. One such case is the use of filters based on Zipf's Law []: these filters, though, require preprocessing large amounts of data in each each relevant language. In addition, their useful application requires the text being filtered to be in a single language and the language correctly detected. All these aspects are prone to error but allow a language-independent system to be built.

In our tests, filtering accuracy was not an issue: this was not a real problem, since we were not interested in the actual output contents, but rather in measuring processing times as a function of input/output sizes.

1.2 Part-of-speech tagging

If counting words is a basic operation in many statistics-oriented approaches, POS tagging is usually the first step in many natural language processing chains. Thus, we also built an application for testing this approach.

Two modules were used: the part-of-speech (POS) tagger (also a morphological analyser) itself [5]; and a morphological disambiguator for marking uninteresting tags according to text [6,4]. A corpus consisting of 3.5 million phrases in Portuguese (taken from news articles) was used to perform the experiment. This corpus was built for studying phenomena in speech synthesis and had already been partially tagged with stand-alone applications. The time taken for the pipelined applications to tag the corpus was about 15 days. We conducted the experiments with the same tools so that times would be comparable and, thus, validate one the approaches (this was the reason not to use SMorph [1,3,4]).

2 Experiments

The first experiment consisted of counting all the n-grams in a corpus with sizes varying from 1 megabyte to 500 megabytes. The experiment counted all the n-grams between two limits, e.g. 1-12 (counting unigrams, bigrams, etc., through 12-grams). The corpus was processed considering various settings to the number of mappers/reducers. The number of active cores was 88. Results are presented in figure 2.

The second experiment considered a POS tagging task on a corpus with sizes varying from 1 megabyte to 500 megabytes. As before, the corpus was processed considering various settings to the number of mappers/reducers. As begThe number of active cores was 88. Results are presented in figure 3.

The numbers on the left of the title (88, 176, 352) correspond to the number of mappers, while the numbers on the right (22, 44, 88) represent the number of reducers. These number translate to the amount of parallel processing available to the task.

3 Final Remarks

Concerning the n-gram counting application, the maximum thoughput was 1.6 MB/s, corresponding to 176 mappers and 88 reducers. Since the mappers produce essentially windowed repetitions of the input, their

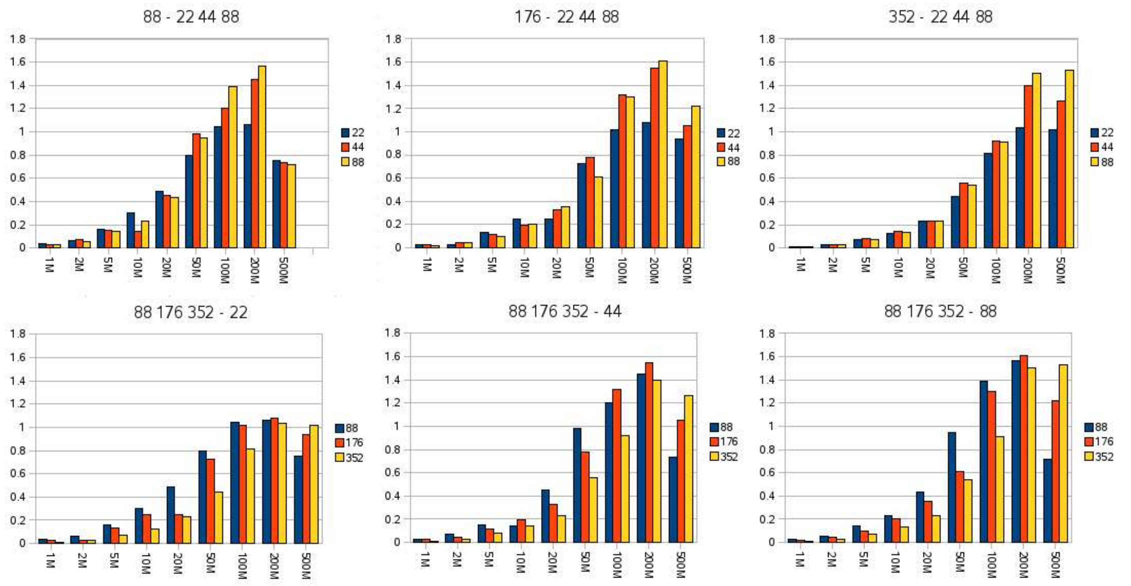

Fig. 2. Throughput results for n-gram counting (MB/s).

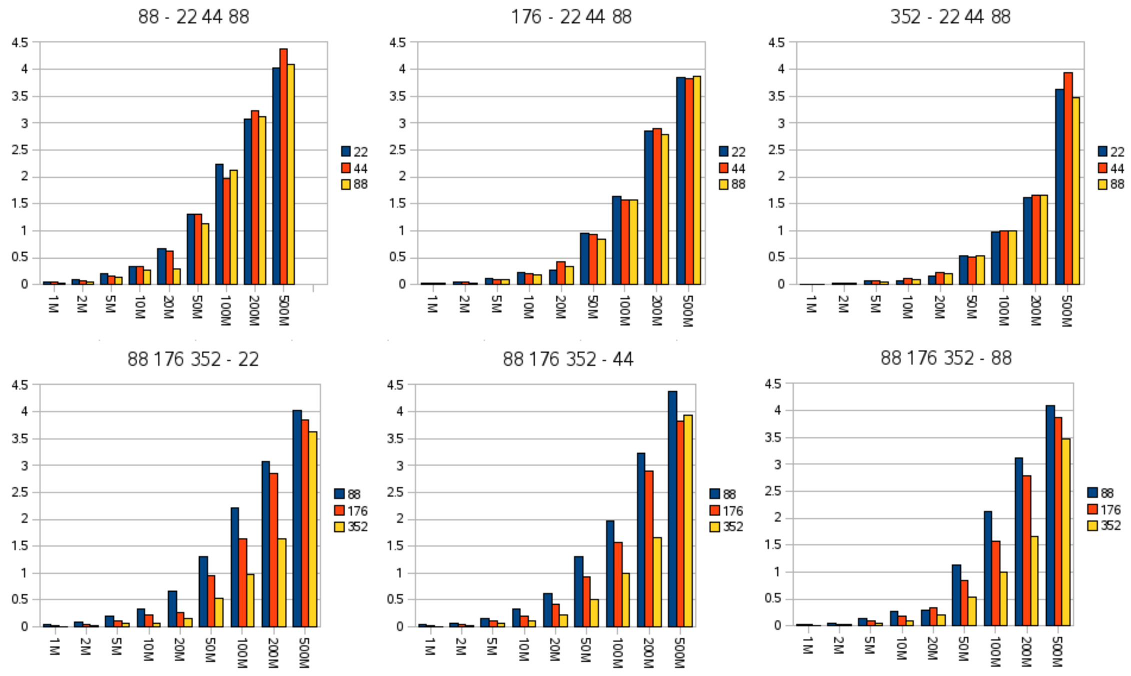

Fig. 3. Throughput results for POS tagging (MB/s).

output is much larger than their input, and increasing the number of reducers also increases throughput. Increasing the number of mappers when the number of reducers is too low (see, for instance, the table 88 176 352 - 22), results in performance degradation.

Concerning the POS tagging application, the results may perhaps be unfair to the original application, but the global time differences cannot be avoided: the time to parse a 450-megabyte corpus using the stand-alone versions of Palavroso and MARv on a single machine was measured in days, while the processing of a 800-megabyte corpus on our cluster, using the NLE-GRID architecture was carried out in about 120 seconds.

Concerning throughput in the POS task, the output from mappers is always approximately the double amount of bytes they receive (they only produce the tags and the original input words). Thus, throughput behaviour is different from the n-gram counting application: now, increasing the number of reducers does not automatically increase performace, while increasing the number of mapper beyond 88 actually causes a decrease in throughput. The best results are obtained by a combination of 88 mappers and 44 reducers.

The behaviour of these two tasks appears to differ significantly because we are only considering throughput and not the actual time taken to run. In the first case, the total run time was approximately 12 minutes, while in the second case, the run time was always less than 3 minutes. The degradation in throughput in the second case can be attributed to the significant overhead in launching helper applications in the mappers.

References

1. S. Aït-Mokhtar. *L'analyse présyntaxique en une seule étape*. Thèse de doctorat, Université Blaise Pascal, GRIL, Clermont-Ferrand, 1998.
2. David Martins de Matos and Tiago Luís and Ricardo Daniel Ribeiro. NLE-GRID T1: Architectural Model (POSC/PLP/60663/2004). Technical Report 30, L^2F – Spoken Language Systems Laboratory, INESC ID Lisboa, Lisboa, Portugal, January 2008.
3. Caroline Hagège. SMORPH: Um analisador/gerador morfológico para o Português. In *Workshop sobre taggers para o Português*, pages 49–74, Lisboa, Portugal, 1997.
4. Luís Marujo, Wang Lin, David Martins de Matos. NLE-GRID T3: Multi-Component Application Builder (POSC/PLP/60663/2004). Technical Report 33, L^2F – Spoken Language Systems Laboratory, INESC ID Lisboa, Lisboa, Portugal, January 2008.
5. J. C. Medeiros. Processamento Morfológico e Correcção Ortográfica do Português. Master's thesis, Instituto Superior Técnico – Universidade Técnica de Lisboa, Lisboa, Portugal, 1995.
6. Ricardo Ribeiro. Anotação morfossintáctica desambiguada do português. Master's thesis, Instituto Superior Técnico, Portugal, 2003.