Marcin Kuta*, Jacek Kitowski**

BENCHMARKING HIGH PERFORMANCE ARCHITECTURES WITH NATURAL LANGUAGE PROCESSING ALGORITHMS

Natural Language Processing algorithms are resource demanding, especially when tuning to inflective language like Polish is needed. The paper presents time and memory requirements of part of speech tagging and clustering algorithms applied to two corpora of the Polish language. The algorithms are benchmarked on three high performance platforms of different architectures. Additionally sequential versions and OpenMP implementations of clustering algorithms were compared.

Keywords: *benchmarking, part-of-speech tagging, document clustering, natural language processing, high performance architectures*

BENCHMARKING ARCHITEKTUR WYSOKIEJ WYDAJNOŚCI ALGORYTMAMI PRZETWARZANIA JĘZYKA NATURALNEGO

Algorytmy przetwarzania języka naturalnego mają duże zapotrzebowanie na zasoby komputerowe, szczególnie gdy wymagane jest dostosowanie algorytmu do języka fleksyjnego jakim jest np. język polski. Artykuł przedstawia wymagania czasowe i pamięciowe algorytmów tagowania częściami mowy oraz algorytmów klasteryzacji zastosowanych do dwóch korpusów języka polskiego. Dokonano benchmarkingu algorytmów na trzech platformach wysokiej wydajności reprezentujących różne architektury. Dodatkowo porównano wersję sekwencyjną oraz implementacje OpenMP algorytmów klasteryzacji.

Słowa kluczowe: *benchmarking, tagowanie częściami mowy, klasteryzacja dokumentów, przetwarzanie języka naturalnego, architektury wysokiej wydajności*

1. Introduction

Part of speech (POS) tagging is a cornerstone of almost any text processing task including parsing, machine translation, information retrieval [9], word sense disambiguation, speech recognition, and many others. Polish is a highly inflective language

* AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, IT and Electronics, Department of Computer Science, al. Mickiewicza 30, 30-059 Krakow, Poland, {mkuta,kito}@agh.edu.pl

** AGH University of Science and Technology, ACC CYFRONET AGH, ul. Nawojki 11, 30-950, Krakow, Poland

and tagging such language is much more difficult than analytic languages like English. One of difficulties is that tagging algorithms are especially computationally time demanding when applied to languages described with large tagsets.

The documents clustering is another important technique applied in Natural Language Processing (NLP), especially in unsupervised organization of documents, returning web search results, creating lexico-semantic nets (wordnets) [1], information retrieval and extraction [2]. POS tagging helps to select from a text sequence the most important features for clustering algorithms.

Works on NLP algorithms usually focus on quality measures, expressed in terms specific to a family of methods (e.g. accuracy for POS tagging, purity for clustering algorithms, etc.). This paper presents different aspect, i.e., time and memory requirements of some of POS tagging and clustering algorithms applied to the Polish language [5, 8]. These requirements are important factors, which can affect the choice of the respective algorithm or make useless the algorithm, which is the best according to the main measure. Intensive resources usage justifies application of high performance architectures to above NLP computations.

The aim of the article is to compare performance of NLP algorithms applied to two corpora of the Polish language on different high performance architectures.

The rest of the paper is organized as follows. Section 2 presents considered algorithms and reasons NLP computations are time intensive. Section 3 describes testbed for experiments. The detailed time and memory performance of selected POS tagging and clustering algorithms is depicted in Section 4. Section 5 concludes the paper.

2. Considered NLP algorithms

The first group of considered experiments was application of various POS tagging algorithms to the Polish language. We examined baseline tagging algorithms and three groups of combined tagging methods.

Seven baseline algorithms were investigated:

- Hidden Markov Model (HMM).
- Maximum entropy method (MET).
- Memory-based learning (MBL).
- Transformation-based error-driven learning (TBL).
- Support vector machines (SVM).
- Decision Trees tagging (DT).
- Conditional random fields (CRFs).

The description of the algorithms and their accuracy for Polish can be found in our work [4].

The combined taggers are not independent taggers and need output of several baseline taggers (called in this context component taggers) both for training and tagging. Output tagging is chosen due to voting procedure, vote strength of each component tagger is established during training phase. Bigger number of component taggers results usually in higher accuracy of a combined tagger but requires more time and memory. Among combined methods we focus on weighted probability distribution voting (WPDV) method [10].

All taggers were evaluated on the modified corpus of Frequency Dictionary of Contemporary Polish (m-FDCP)[1] [5]. Two versions of the tagset were taken into account: the full version, referred to the complex tagset; and the reduced version, carrying only basic information about part of speech, referred to the simple tagset.

The second group of experiments, i.e. clustering experiments, was conducted on the set of 10,000 articles selected from one of the main Polish newspapers, *Rzeczpospolita* (henceforth the ROL corpus)[2]. The aim of the experiment was to find the best clustering algorithm among four algorithms: agglomerative, direct, repeated bisections (RB), and repeated bisections with refinements (RBR) [11]. The number of clusters was equal to six and details of the clustering experiment are given in [6].

2.1. Tuning NLP algorithms to Polish

For the POS algorithms the multiple or time intensive computations were required due to the following issues:

- Parametric optimization of taggers. The default settings of taggers, accommodated often for English, need not to be the best choice for Polish.
- Effect of training data size (analysis of learning curves). The bigger the training corpus the higher the precision of a tagger, but this is bound to higher training time. The detailed influence of size of a training corpus on precision of a tagger is discussed in [4].
- Training combined taggers with n-fold cross validation. The problem arises for this type of taggers, as they need for the training process a tuning corpus, separate from the training corpus of baseline taggers. In our works 9-fold cross validation was always applied.
- The number of baseline taggers used in composition of combined taggers [7]. Adding a new tagger as a component accounts to significant decline of tagging speed of a combined tagger. The problem arises mainly for the WPDV combined taggers.

The following aspects of documents clustering algorithms and their tuning were considered:

- Different representations of the corpus (feature selection). The features are selected on the basis of the part of speech of a token and a number of consecutive tokens. Several representations of the corpus were taken into account:
  - base representation – noun, verb and adjective tokens are represented, while tokens with other part of speech are discarded,

[1] m-FDCP corpus is available from `http://nlp.icsr.agh.edu.pl`

[2] ROL corpus is available from `http://www.cs.put.poznan.pl/dweiss/rzeczpospolita`

  - noun representation – only noun tokens are taken into representation of the corpus,
  - bigram representation – base representation is enriched with presence of bigrams,
  - trigram representation – bigram representation is enriched with presence of trigrams.

- Different term weighting schemes. The most popular term frequency-inverse document frequency (*tfidf*) scheme and slightly less popular logarithmic entropy (*logent*) scheme were considered.
- Alternative between the Vector Space Model (VSM) and the Latent Semantic Analysis (LSA) model. The VSM model represents each document as a vector and each dimension corresponds to a separate term. Within the LSA model each dimension corresponds to a semantic concept. The LSA space is computed from the VSM space by the Singular Value Decomposition (SVD) of a document-term matrix.
- Investigation of the optimal number of dimensions, in case of the LSA model. The number of dimensions varied from 50 to 5000 in our experiments.
- Several cluster criterion functions, driving the process of building optimal clusters [11]. Twelve criterion functions were applied, seven of them to partitional algorithms and ten to the agglomerative algorithm.
- Multiple execution of clustering algorithms relying on random initial clustering, i.e., direct and repeated bisection algorithms. Multiple runs showed average quality of the above algorithms.

3. Experimental testbed

Experiments were performed on the following three platforms:

- The first platform was the SGI Altix 3700 (SMP architecture), equipped with 256 1.5 GHz Intel Itanium 2 CPUs, 512 GB shared memory and 7.75 TB disk storage. Processors implemented IA-64 architecture.
- The second platform was a IBM BladeCenter HS21 computing cluster consisting of 56 nodes, each node containing two Intel Xeon Dual Core 5150 2.66 GHz processors and 8 GB RAM. Totally, the cluster provides 112 processors, 448 GB RAM, and 5 TB disk storage. Each CPU contains 4096 KB cache memory shared between two cores. The cluster achieves 1192 Gflops theoretical computing power. Processors implemented x86-64 architecture.
- The last tested platform was a SMP machine equipped with four dual core AMD Opteron 865 1.8 GHz processors and 20 GB shared main memory. Each core contains 1024 KB cache memory. Processors implemented x86-64 architecture.

The computational resources were shared between members of scientific community and only small fraction of resources was available for our tasks.

For tagging we used gcc compilation with standard compiler options and also Perl and Java interpreters. The clustering was performed with the CLUTO toolkit [3] distributed as binary version, compiled with gcc (sequential version) or Intel C++ 8.1 compiler (OpenMP version). The distributions for SGI Altix 3700 and OpenMP version were available as 32-bit versions only (to be used in compatibility mode), while distributions for HS21 and SMP Opteron were full 64-bit versions.

4. Results

In experiments we used job level parallelism with one core allocated to one sequential process. When benchmarking clustering algorithms each job consisted from a group (sequence) of clustering algorithms executed for one model of the ROL corpus (e.g. bigram model) and for one, chosen number of dimensions in LSA space (e.g. 1000). A group of clustering algorithms was executed sequentially within a job. The clustering algorithms within a job differed in their nature (partitional vs. agglomerative) and by applied criterion functions. To speedup computations, jobs for different models of the ROL corpus or different number of LSA dimensions were executed in parallel on separate cores.

The only exception were OpenMP computations, where four cores were used in parallel by one job.

Table 1
CPU time and memory requirements of an instance of a baseline tagger trained on 90% of the m-FDCP corpus (SGI Altix 3700)

Tagger	Training time [s]	Tagging speed [tokens/s]	Memory usage [GB]
	Simple tagset		
HMM	2	25 100	0.02
MET	1400	120	0.20
TBL	3000	1500	0.32
MBL	80	14 100	0.11
SVM	16 600	520	0.86
DT	5	3700	0.03
CRF	54 500	9200	0.70
	Complex tagset		
HMM	4	12 300	0.02
MET	40 700	13	0.31
TBL	60 800	1100	1.11
MBL	100	8700	0.15
SVM	78 900	170	1.77
DT	400	1100	0.05

Table 2
CPU time and memory requirements of an instance of a baseline tagger trained on 90% of the m-FDCP corpus (IBM HS21)

Tagger	Training time [s]	Tagging speed [tokens/s]	Memory usage [GB]
	Simple tagset		
HMM	1	146 500	0.004
MET	567	2200	0.08
TBL	783	4600	0.29
MBL	31	21 800	0.14
SVM	2600	1100	0.87
DT	1	183 100	0.01
CRF	48 700	14 600	4.10
	Complex tagset		
HMM	1	52 300	0.01
MET	12 800	80	0.08
TBL	11 400	4200	1.13
MBL	58	4600	0.13
SVM	22 600	400	1.17
DT	123	44 800	0.03

Time requirements of considered algorithms (training time, tagging speed, clustering time) are expressed with help of CPU time and memory usage stands for physical memory usage. Tables 1–4 concern sequential versions of presented algorithms while Tables 5–6 concern parallel OpenMP version.

Tables 1–3 provide training time, tagging speed, and memory usage of baseline taggers trained on 90% of the m-FDCP corpus on three architectures described in Section 3: SGI Altix 3700, HS21 and SMP Opteron, respectively. The taggers differ in their performance – from the fastest and using least memory HMM tagger to the slowest and memory consuming CRF tagger.

Table 4 presents training time, tagging speed and memory usage of the WPDV tagger as a function of the number of component taggers. The results reveal unusual feature of the WPDV tagger – tagging time is bigger than training time. In fact, in the case of the complex tagset, tagging time strongly dominates training time. We observe also exponential growth of the tagging time with the increasing number of the baseline taggers constituting the WPDV tagger. Memory usage of the WPDV tagger increases when moving from the simple to the complex tagset. The impact of the number of component taggers on memory usage is observed for the simple tagset.

Time and memory requirements of the algorithms applied to documents clustering of the ROL corpus are presented in Figures 1–5. The corpus preparation algorithm creates input data required by clustering algorithms and includes building

Table 3
CPU time requirements of an instance of a baseline tagger trained on 90% of the m-FDCP corpus (SMP Opteron)

Tagger	Training time [s]	Tagging speed [tokens/s]
	Simple tagset	
HMM	1	75 800
MET	748	3 200
TBL	1000	3 000
MBL	38	17 700
SVM	4400	700
DT	2	63 400
CRF	86 200	9 900
	Complex tagset	
HMM	2	31 700
MET	18 300	150
TBL	17 300	3100
MBL	55	4400
SVM	40 200	200
DT	268	18 600

Table 4
CPU time and memory requirements of the WPDV tagger versus number of component taggers (SGI Altix 3700)

Number of component taggers	4 best taggers	5 best taggers	6 best taggers
	Simple tagset		
Training time [s]	23	20	22
Tagging speed [tokens/s]	2747	749	283
Training – memory usage [GB]	0.46	0.25	0.46
Tagging – memory usage [GB]	0.87	1.82	7.00
	Complex tagset		
Training time [s]	27	30	32
Tagging speed [tokens/s]	14.58	5.69	2.49
Training – memory usage [GB]	1.02	1.19	1.33
Tagging – memory usage [GB]	11.28	11.28	11.33

the document-term matrix in the VSM space, its further SVD transformation to the LSA space and normalisation. The SVD factorization constitutes the main part of the preparation algorithm. Time and memory required by this preparation algorithm is

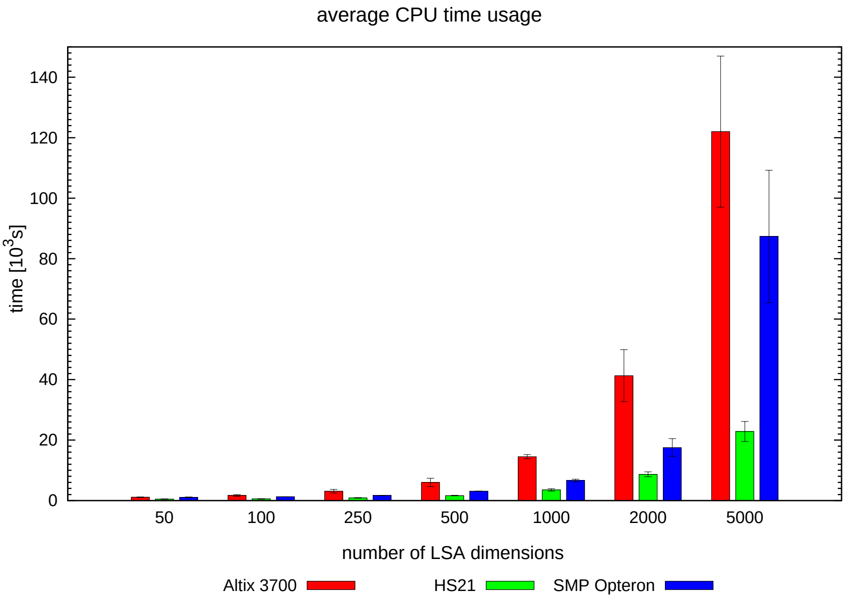

Fig. 1. Average CPU time usage of the ROL corpus preparation algorithm versus number of LSA dimensions (Altix 3700, HS21, SMP Opteron)

given in Figures 1 and 2. Once fixed the number of dimensions of the LSA space, the preparation algorithm was run several times, each time using different representation of the corpus (e.g. base, noun, logent representation, etc.). The resource requirements presented in Figures 1–2 are averaged over these several runs, and their standard deviation is also given. Requirements of the clustering algorithms themselves are shown in Figures 3 and 4. Entries in Figure 3 provide total time of execution of all investigated clustering algorithms (four algorithms, twelve cluster criterion functions) applied to one representation of the corpus in fixed LSA space and Figure 4 shows peak memory

Table 5
Walltime of execution of clustering algorithms [s]

Method	Criterion function	Altix	HS21	HS21 openMP	Opteron	Opteron openMP
RB	$\mathcal{I}_2$	804	142	145	274	300
RBR	$\mathcal{I}_2$	857	145		282	
Direct	$\mathcal{I}_2$	614	97		187	
Agglo	$\mathcal{I}_2$	452	114		180	
Agglo	$\mathcal{H}_1$	28871	13859	3440	24919	5982
Agglo	$\mathcal{H}_2$	32184	18433	3114	26917	5596

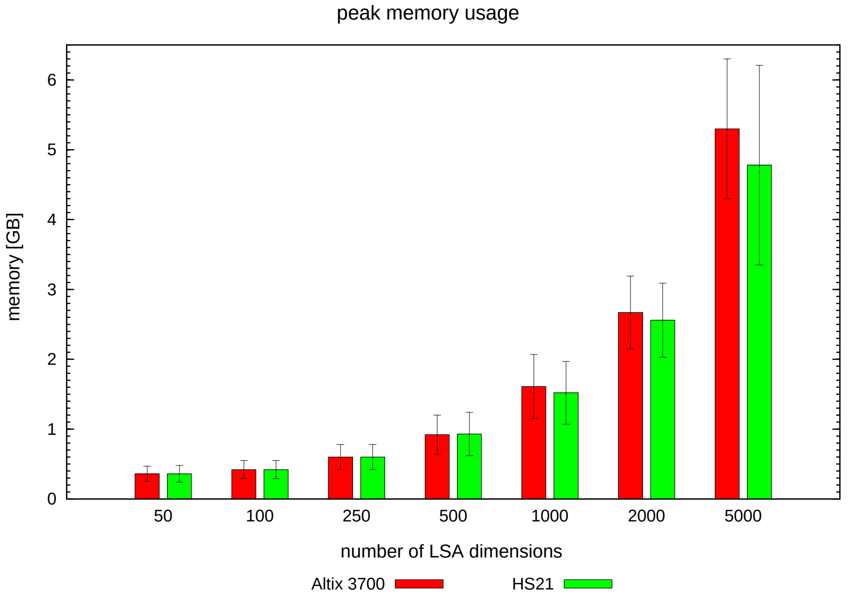

Fig. 2. Peak memory usage of the ROL corpus preparation algorithm versus number of LSA dimensions (Altix 3700, HS21)

usage. Taking into account that run of each direct and bisection algorithm was repeated 10 times to average their results, each group of clustering algorithms consists of 100 runs of algorithms. As for Figures 1 and 2, the results in Figures 3–6 and Table 5 are averaged over runs on different representations of the corpus transformed to the LSA space of fixed number of dimensions. We can observe in Figure 1 that time of corpus preparation grows approximately linearly with increasing number of LSA dimensions. The linear dependence holds also for memory usage versus number of LSA dimensions. Contrary to the preparation algorithm, both time and memory requirements of clustering algorithms remain nearly constant, independent of the LSA space inside which clustering is performed and number of dimensions (Figures 3 and 4).

Figure 5 presents walltime and total CPU time requirements of the group of clustering algorithms. OpenMP enabled versions of these algorithms were applied, wherever possible. The OpenMP versions were run with four CPU cores per process. Among clustering algorithms RB algorithm and agglomerative algorithm with Zhao's $\mathcal{H}_1$ and $\mathcal{H}_2$ criterion functions were OpenMP aware. Total CPU time means the sum of CPU time consumed by four cores used by the group of clustering algorithms. Table 5 shows time requirements of the this experiment from another perspective – agglomerative algorithm with $\mathcal{H}_1$ and $\mathcal{H}_2$ criterion functions dominates execution of the whole group, and speedup of the their OpenMP versions is approximately equal to four, i.e, number of cores single algorithm was running. Figure 6 gives peak memory usage for the last experiment.

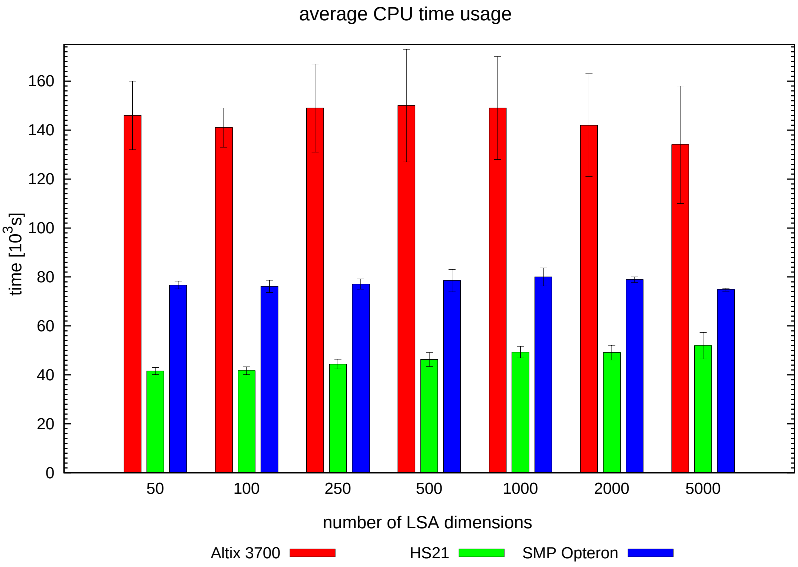

Fig. 3. Average CPU time usage of the group of algorithms applied to documents clustering of the ROL corpus versus number of LSA dimensions (Altix 3700, HS21, SMP Opteron)

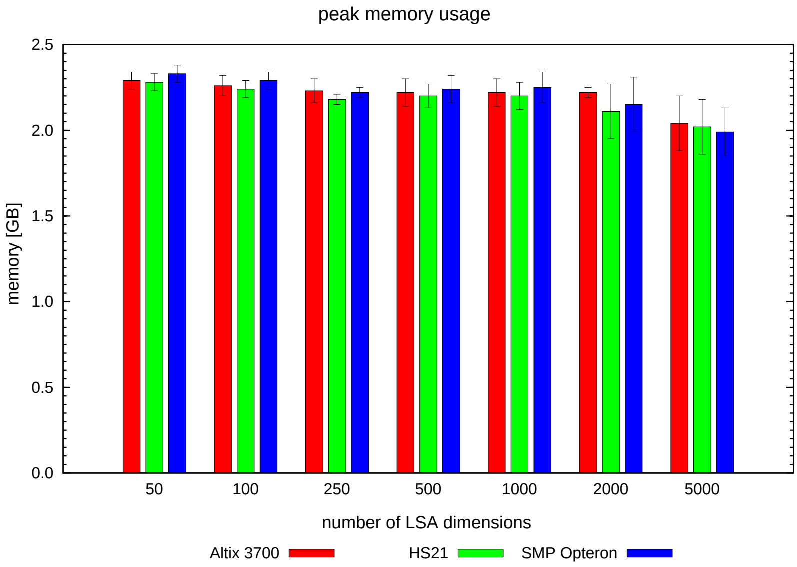

Fig. 4. Peak memory usage of the group of algorithms applied to documents clustering of the ROL corpus versus number of LSA dimensions (Altix 3700, HS21, SMP Opteron)

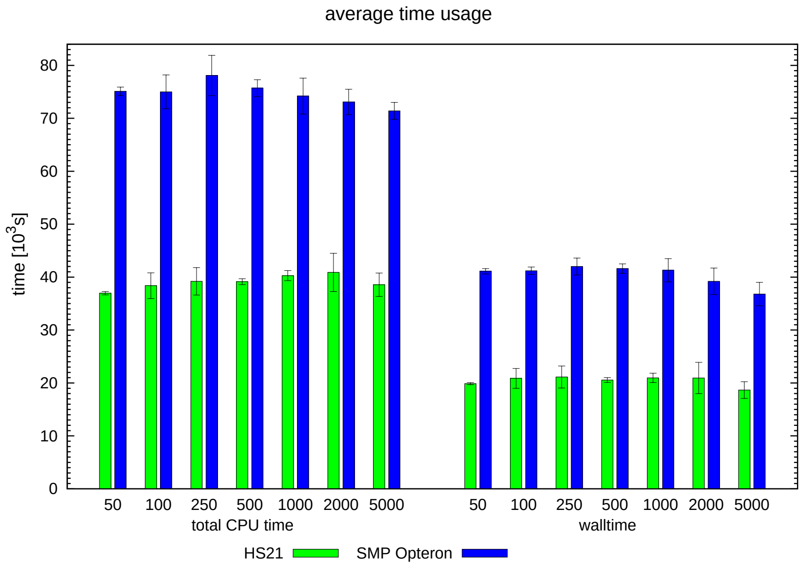

Fig. 5. Average time usage of the group of algorithms applied to documents clustering of the ROL corpus versus number of LSA dimensions (32-bit mode, OpenMP, SMP Opteron and HS21)

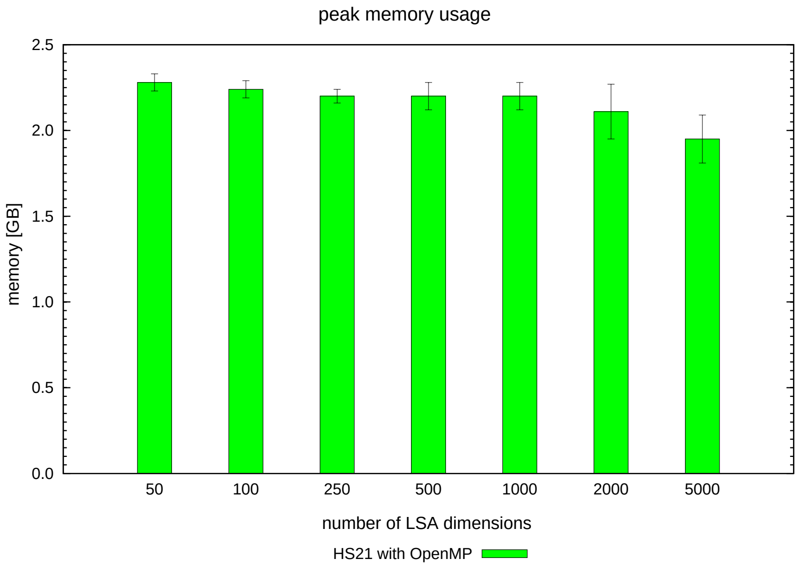

Fig. 6. Peak memory usage of the group of algorithms applied to documents clustering of the ROL corpus versus number of LSA dimensions (32-bit mode, OpenMP, HS21)

5. Conclusions

Some of POS tagging and clustering algorithms are time and memory consuming but tuning above algorithms to Polish requires consideration of many parameters and multiple, even more intensive computations. The high-throughput architectures are well suited to such multivariant computations and satisfy well time and memory demands.

The big advantage of SGI Altix 3700 and HS21 machines was availability of many CPUs, what allowed to run many tasks concurrently. The x86-64 architecture turned out however to be more effective than the Itanium 2 architecture. Among two implementations of x86-64 architecture Intel Xeon was more robust than AMD Opteron, what may be caused by two times bigger size of cache per core for the former one. Taking additionally into account considerably higher failure frequency rate for SMP architectures than for cluster architectures, IBM BladeCenter HS21 turns out the most suitable for our experiments out of three considered high performance architectures.

Acknowledgements

This work was supported by the AGH University of Science and Technology grant no. 10.10.120.865. ACC CYFRONET AGH-UST is acknowledged for granting computational resources of SGI Altix 3700 and IBM BladeCenter HS21 platforms. Prof. K. Boryczyko and M.Sc. R. Górecki are acknowledged for help with access to SMP Opteron machine.

References

[1] Broda B., Piasecki M.: *Experiments in clustering documents for automatic acquisition of lexical semantic networks for Polish.* [in:] Proc. of the 16th International Conference Intelligent Information Systems, Zakopane, Poland, 2008, pp. 203–212.

[2] Piskorski J., Homola P., Marciniak M., Mykowiecka A., Przepiórkowski A., Woliński M.: *Information extraction for Polish using the SProUT platform.* [in:] Proc. of the International Conference Intelligent Information Systems (IIS 2004), Siedlce, Poland, 2004, pp. 227–236.

[3] G. Karypis.: *CLUTO. A clustering toolkit.* Technical Report 02–017, University of Minnesota, Department of Computer Science, 2003.

[4] Kuta M., Chrząszcz P., Kitowski J.: *A case study of algorithms for morphosyntactic tagging of Polish language.* Computing and Informatics, 26(6), 2007, pp. 627–647.

[5] Kuta M., Chrząszcz P., Kitowski J.: *Increasing quality of the Corpus of Frequency Dictionary of Contemporary Polish for morphosyntactic tagging of the Polish language.* Computing and Informatics, 28(3), 2009, pp. 319–338.

[6] Kuta M., Kitowski J.: *Clustering Polish texts with latent semantic analysis.* [in:] Proc. of the 10th International Conference on Artificial Intelligence and Soft Computing, Zakopane, Poland, 2010, pp. 532–539.

[7] Kuta M., Wójcik W., Wrzeszcz M., Kitowski J.: *Application of stacked methods to part-of-speech tagging of Polish.* [in:] Proc. of the 8th International Conference on Parallel Proc. and Applied Mathematics, Wrocław, Poland, 2009, pp. 340–349.

[8] Kuta M., Wójcik W., Wrzeszcz M., Kitowski J.: *Application of weighted voting taggers to languages described with large tagsets.* Computing and Informatics, 29(2), 2010, pp. 203–225.

[9] Radovanović M., Ivanović M., Budimac Z.: *Text categorization and sorting of web search results.* Computing and Informatics, 28(6), 2009, pp. 861–893.

[10] Halteren H. van, Zavrel J., Daelemans W.: *Improving accuracy in word class tagging through the combination of machine learning systems.* Computational Linguistics, 27(2), 2001, pp. 199–229.

[11] Zhao Y., Karypis G.: *Hierarchical clustering algorithms for document datasets.* Data Mining and Knowledge Discovery, 10(2), 2005, pp. 141–168.