Computer Science • 13 (4) 2012 http://dx.doi.org/10.7494/csci.2012.13.4.5

Štefan Dlugolinský
Martin Šeleng
Michal Laclavík
Ladislav Hluchý

# DISTRIBUTED WEB-SCALE INFRASTRUCTURE FOR CRAWLING, INDEXING AND SEARCH WITH SEMANTIC SUPPORT

**Abstract** *In this paper, we describe our work in progress in the scope of web-scale information extraction and information retrieval utilizing distributed computing. We present a distributed architecture built on top of the MapReduce paradigm for information retrieval, information processing and intelligent search supported by spatial capabilities. Proposed architecture is focused on crawling documents in several different formats, information extraction, lightweight semantic annotation of the extracted information, indexing of extracted information and finally on indexing of documents based on the geo-spatial information found in a document. We demonstrate the architecture on two use cases, where the first is search in job offers retrieved from the LinkedIn portal and the second is search in BBC news feeds and discuss several problems we had to face during the implementation. We also discuss spatial search applications for both cases because both LinkedIn job offer pages and BBC news feeds contain a lot of spatial information to extract and process.*

**Keywords** distributed web crawling, information extraction, information retrieval, semantic search, geocoding, spatial search

## 1. Introduction

Due to the fact that web content is very diverse, building an intelligent web-scale search service is still a challenge and faces plenty of problems. There are many heterogeneous sources of information in different languages and there are many different formats of information representation too. According to latest surveys [29], there are about 65% of top 1 million websites using XHTML markup language, while the rest use HTML. The trend of XHTML usage is slightly growing, but on the other hand, semantic standards like RDF, GRDDL, RDFa, SPARQL, OWL, RIF or SKOS occur sporadically in XHTML. In general, Semantic Web solutions based on mentioned standards cannot be yet applied. Therefore, there is still a need to deal with the information extraction and semantic analysis of crawled web documents to support intelligent search. We propose an architecture for distributed large-scale information processing for the intelligent web search task in the following chapter.

## 2. Architecture

An important aspect of the web-scale search service is its scalability. It can be fulfilled by distributed architecture [18, 14, 2], which can process a large amount of data in parallel. Our search service is built on top of Apache Hadoop [25], an open-source implementation of MapReduce paradigm introduced by Google in 2004 [3].

The benefits of using the MapReduce architecture are discussed in our prior work [15], where we compare its performance to a single machine solution. There are two different applications compared with the result of 1.9 and 12 times performance gain after executing them on a Hadoop cluster.

The system presented in this paper operates over HDFS (Hadoop Distributed File System) [24]. Some of the produced data is stored in HBase [26] (open-source implementation of the Google BigTable [1]). In general, there can be any distributed file system used in Hadoop under the condition that it can be mounted by the underlying operating system. Internet content is crawled and processed by Apache Nutch [27], which is executed in the Hadoop distributed environment in parallel. Nutch is a very powerful crawler with a lot of configurable features, plug-ins and filters such as:

- domain filter based on regular expressions to target crawler on particular domains,
- regular expression URL filter for URL parameter filtering,
- document type filter to refine, which types of document to crawl,
- dynamic URL filter for dynamically generated web pages,
- several plug-ins for different kind of content, format, etc.

In order to perform our crawling tasks over BBC feeds and LinkedIn job offers, several Nutch plug-ins were customized and implemented:

- customized built-in HTML parser to preserve visual formatting of the document textual content for better information extraction,

- new information extraction plugin based on regular expressions and gazetteers,
- new specialized indexing plug-in for indexing entities extracted by information extraction plug-in,
- new specialized plugin for geocoding,
- new specialized geo-location indexing plug-in.

The search part of our framework is based on Apache Solr [22] and Apache Velocity [23] projects. Apache Solr is a subproject of Lucene framework and it extends its search capabilities. Apache Velocity permits one to use a simple but powerful template language to reference objects defined in Java code and thus implement user interface comfortably. The proposed architecture of the search system is shown in Figure 1.

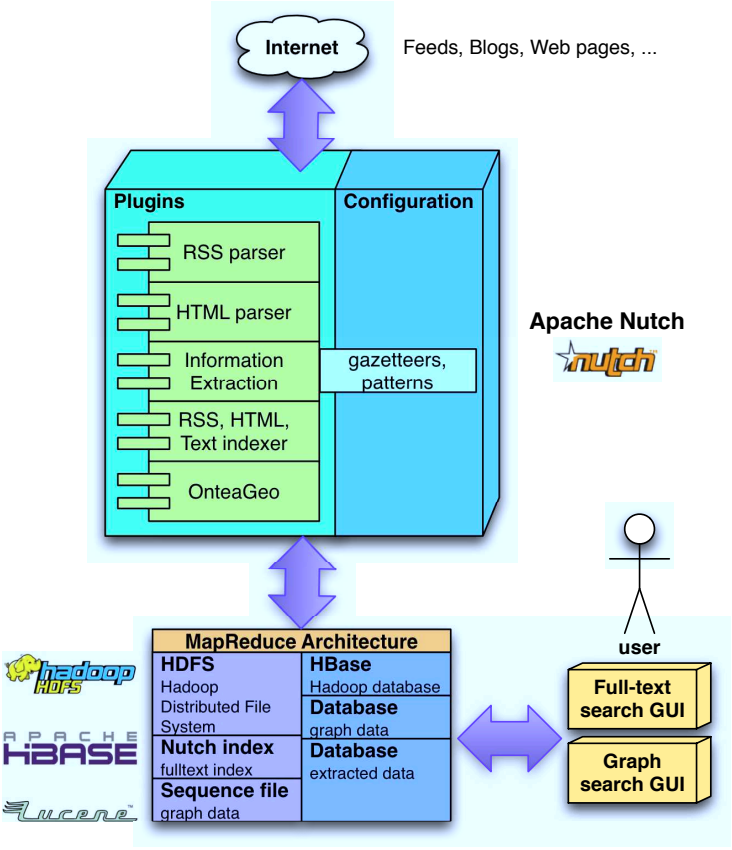


**Figure 1.** Architecture of the distributed web-scale search system.

Users can use facets when performing full-text search. Facets refer to categories used to characterize information items in a collection, thus when a label within a facet

is selected, all items that have been assigned to that facet are retrieved [11]. Information items in a collection are represented by retrieved documents and facets are built from extracted entities, e.g. Person, Country. Facets in Solr are treated as flat and non-hierarchical, so it is very simple to define them. Facets are built automatically from entities extracted from documents, where each entity type specifies a facet category. Properly extracted information used to build facets can bring great value to the navigation because documents can be categorized according to their semantic content. A similar approach of building facets is used in [10] where authors extract semantic information from Wikipedia using the DPpedia Framework.

There is also a graph search user interface available, which allows users to search within a graph built from extracted entities. A spread activation algorithm is applied during the graph search.

As it is stated earlier in the text, there is only LinkedIn website [16] crawled for job offers currently. In the future we want to crawl and parse also Monster website [17] to retrieve job offers and user CVs. LinkedIn claims it offers roughly 80 000 job offers (79 871 at the time of making our tests — October 10, 2011). There are 70 116 job offers crawled in our database with an overall size of 2.1 GB crawled/parsed data and around 740 MB metadata with the index (includes also textual content).

We must mention that we did not start from scratch in this work, but we exploited several tools developed in the scope of the NAZOU project [19] (e.g. Ridar and Erid) [6].

The system was deployed and tested on a small cluster with 1 server and 8 nodes with configuration described in Table 1. Overall storage capacity is over 5 TB and every node in the cluster can process up to 6 tasks in parallel (48 for whole cluster).

**Table 1**
Cluster node configuration.

| | |
|---|---|
| Processor | Intel®CoreTM 2 Quad CPU Q9550 2.83GHz |
| System memory | 4 GB |
| Storage | WDC WD7500AACS-0 (750 GB) |
| OS | Linux 2.6.24-19-generic x86_64 GNU/Linux |

## 3. Information extraction

Various information is extracted from the textual content of crawled web documents as well as from the HTML DOM objects. A built-in Nutch html-parser plug-in is modified to produce formatted output of the textual content found in a web document. This parser tries to preserve visual formatting of the source HTML page in the output text, so this feature can be exploited in the text segmentation and better information extraction. Simple gazetteers, regular expression patterns and a combination

of both approaches are used for information extraction. There are several types of NEs (Named Entities) are extracted. The entities listed below were extracted from the LinkedIn job offers:

- job posted date,
- job offering company,
- industry in which the job is offered,
- location related to the job offer and offering company (i.e. JobLocation, City and Country),
- required skills for the offered position,
- expected experience of the applicant,
- generic named entities.

The following entities are extracted in the BBC news task:

- PersonName,
- TelephoneNumber,
- location related entities Address, City and Country,
- generic name entities.

Person names are extracted in two-step approach, where in the first step a gazetteer is used to match a given name and finally extraction patterns with gazetteer results awareness are applied. The same approach is used for other NE types (e.g. CompanyName, TelephoneNumber). There are also patterns used, which combine results extracted by other patterns. This way an Address entity is extracted for example. The generic named entities are sequences of capital letter starting words (with sentence beginning awareness) and have assigned a "NE" key (e.g. NE ⇒ "Gare SNCF"). Users can interact with the extracted data using the graph search tool and change the type of extracted entity, delete its value or merge entities representing the same object (e.g. "International Monetary Fund" with "IMF"). This way the user can help in creating negative and positive gazetteers for the next re-parsing and extraction.

Location related entities (geo-entities) extracted from web documents are in the form of a free-form text and need to be converted into latitude/longitude coordinates before they can be used for indexing the documents and in spatial search. The conversion process is called geocoding. There are several free geocoding services available. The most known are Google Geocoding [7] and Yahoo! PlaceFinder [30] services. We use both services as a basis for our geocoding approach, which is explained in more detail in chapter 6.2 on LinkedIn task. After the geo-entities are extracted and geocoded, they are ready for indexing. The spatial indexing is described in chapter 4.1.

Generic NEs are used also for creating JobSkill gazetteers. This is done by taking NEs extracted from the "Desired Skills & Experience" part of job offer and then filtering out those with the lowest frequency. Finally a gazetteer list is built and applied in the next crawl cycle. Since there is no strict job offer structure required by LinkedIn, some job offers have their own structure, but most of the job offers contain recommended parts like the mentioned one.

## 4. Indexing

Fetched web documents are indexed by all extracted entities described in chapter 3. These entities are used inside the Lucene index as fields (following the key/value sense). If there are multiple entities of the same type but with different values extracted, they are all put into the index, because multi-valued fields are supported in Nutch (since version 1.2).

### 4.1. Spatial indexing

The concept of indexing by spatial data is very important. If talking about web documents, there are two general approaches to indexing by geographic coordinates. The first is to index each document by one geographic location and the second is to index each document by multiple geographic locations. The advantage of the first method is in straightforward indexing and searching implementation, where each document in the index has been assigned a pair of latitude/longitude coordinates. Therefore, searching is a simple question whether a tested document's latitude/longitude coordinates are within a specified range. The disadvantage is that only one geographic location can be assigned to each document in the index. The second approach is more suitable for indexing web documents, because it is natural that one document could refer to more geographic locations and it is expected then to index such document by all of them. For instance a news article, which informs about explosions in two different cities or a job offer, where several positions on different places can be announced.

The first indexing approach is currently implemented in Apache Lucene [21] (Lucene is an indexing base for Solr and Nutch), but there are also other methods investigated in Lucene, which follows the second approach [28]. To be more precise, there is CartesianTier concept, which has been abandoned and LocalLucene, which is still under development. The methods of the second approach exploit hash functions to encode latitude/longitude coordinates into a single string, which gives an ability to store coordinates in a multi-valued index field and to attach multiple geo-locations to one document.

In our previous work we have showed a suitable indexing (as well as searching) approach which uses an HTM (Hierarchical Triangular Mesh) [12, 20] method for indexing geo-locations on the Earth's surface. We integrated it and tested for Nutch 0.9 [5, 4]. In this work, we use the same spatial indexing approach and integrated it in Nutch 1.3 and Solr 3.1. There is an HTM ID computed for each geocoded geo-entity (represented by latitude and longitude) and stored in the "GeoHash" index field of particular document. The GeoHash field is then used in spatial searches.

# 5. Search

## 5.1. Full-text search with facets

A full-text search with facets is accessible by customized native Apache Solr user interface, where users can input their queries and receive displayed results (see the Fig. 2 and Fig. 3). On the left side of the pane we can see several lists of the 10 most frequent values for each indexed entity type (JobTitle, JobCompany, JobLocation, etc.) — "Field Facets". In the top pane there is a "Find" search box (for full text search) with the list of already selected facets under it. As an example of a full-text faceted search, we can search for "php" to filter job offers with the word "PHP" in the content. We receive 2 944 results. If we are interested only in full-time jobs and a requirement for MySQL and JavaScript, we select corresponding facets and receive 203 filtered job offers. The search can be also restricted to London. Then 10 results matching the criteria for the London area are returned. A faceted search is also connected with entity relation search tool gSemSearch [13], which benefits of entity relation graph traversing and spreading activation.

Lucene/Solr dispose with a rich query language, which interprets query strings into a Lucene query. One of such query types are range queries, which are used inside our application to filter jobs by their submission date. Range queries can be applied on custom numerical fields like "salary" field, for instance.

## 5.2. Spatial search

Spatial search can be performed in several ways depending on the method of spatial indexing. The method of spatial indexing being used in the system gives several advantages. One can perform bounding-box queries, circle radius queries or any bounding-shape queries. The idea is to pre-compute geo-hash prefixes for a search area and then test in-index geo-hashes on a prefix match. If a tested document's geo-hash matches the search area prefix, the document is considered to be inside the search area. More details are discussed in [4]. A navigable map has been added to the user interface to easily specify search interest bounding-box. Full-text search, like it is described in chapter 5, can be easily used together with the spatial search.

# 6. Experiments

## 6.1. BBC News and RSS feeds

BBC news contains a lot of interesting information about what is happening in the world. There are 18 705 web pages crawled, including RSS feeds and there are extracted entities like person names, addresses, cities, countries and NEs. Spatial entities are geocoded and indexed so that the documents in which they appear can be found when searching within a bounding-box.

More details on BBC index are available in Table 2. Documents represent BBC news. Doc. ratio stands for percentage of overall fetched documents, where particular


**Figure 2.** Faceted search in Apache Solr.

entity occurs at least one time. Totally 1 501 distinct geo-entities (i.e. Address, City and Country) are extracted from the BBC news. LatLon field stores geocoded coordinates and GeoHash field is indexed field with HTM ID value (see chapter 4.1).

There is a user interface available at `http://try.ui.sav.sk:7070/apache-solr-3.1.0/browse` (Fig. 3), but it is still experimental and not very user friendly. We use it only for testing purposes.

## 6.2. LinkedIn job offers

There are LinkedIn job offer pages crawled in this experiment and important information related to a particular job offer is extracted. Job offer pages need to be crawled periodically since they are updated and outdated. User interface for job offer search is available at `http://try.ui.sav.sk:7070/2012-01-07/browse`.

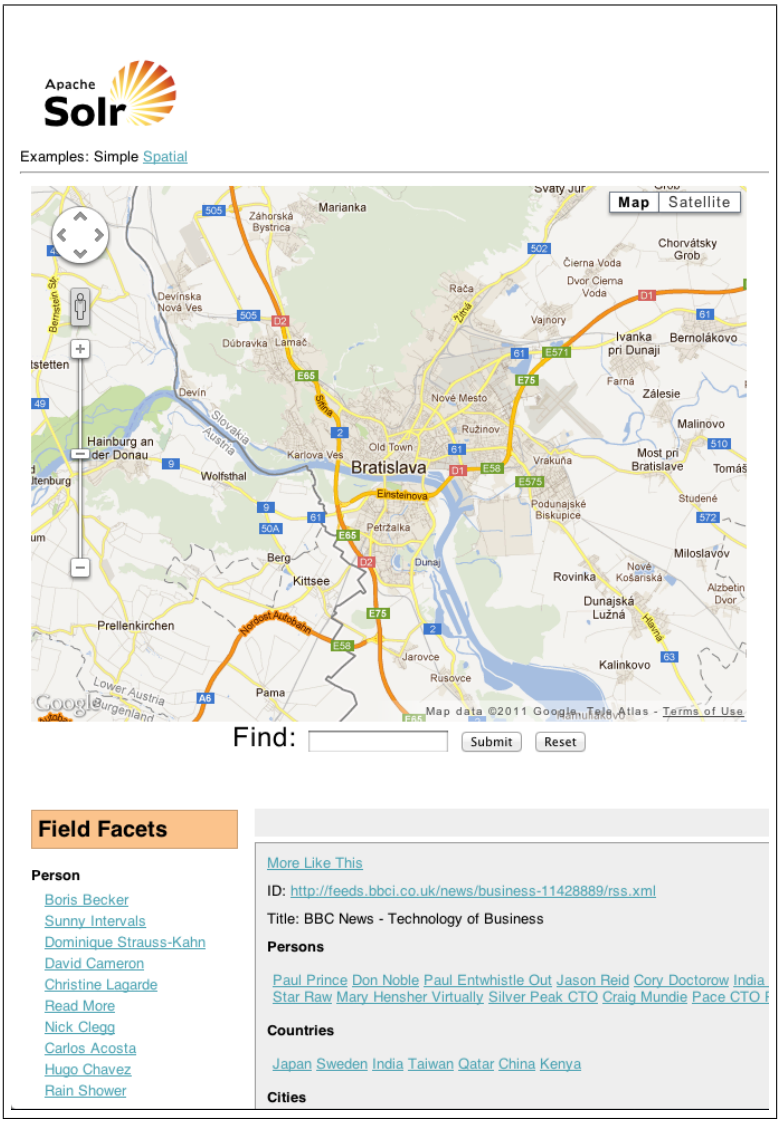


**Figure 3.** Spatial search enhancement in Apache Solr.

More detailed information about one of our test crawls for LinkedIn is available in Table 3. There are 70 116 job offers of overall 113 268 LinkedIn documents fetched, which is about 62%. Doc. ratio represents the percentage of all fetched documents, where a particular entity occurs at least one time. The percentage marked with an asterisk is computed from the total number of job offer pages since the corresponding entity is extracted only from job offer pages.

**Table 2**
Index statistics for the BBC task.

| | | | |
|---|---|---|---|
| Documents | | 18 705 | |
| Terms | | 520 625 | |
| **Entity type** | **Docs** | **Doc. ratio** [%] | **Distinct** |
| Address | 44 | 0.24 | 62 |
| City | 12 933 | 69.14 | 1 152 |
| Country | 11 082 | 59.25 | 287 |
| LatLon | 16 101 | 86.08 | 2 167 |
| GeoHash | 16 101 | 86.08 | 2 145 |
| NE | 18 698 | 99.96 | 94 936 |
| Person | 17 467 | 93.38 | 39 355 |
| TelephoneNumber | 3 | 0.02 | 10 |
| Title | 18 705 | 100.00 | 15 133 |

The system extracts non-spatial information such as JobTitle, JobType, JobFunction, Company, Industry, Skill, Experience, PostedDate and spatial information like JobLocation, City and Country. City and Country entities are extracted by the gazetteer from the textual content only (just like it is in the BBC news task), while the JobLocation is extracted by traversing the HTML DOM tree of a job offer page and looking for its DIV element. City and Country entities are geocoded directly, while the JobLocation entities need to be treated differently, because of their content. JobLocations are in a free form text, always beginning with a company name followed by a location of the job. There is no strict format for the location. It can be anything that user writes down, for example "Anywhere" as it has been seen many times in offers. Below are some concrete examples of such JobLocations:

- Pegasystems Inc. — Anywhere (Austin, Texas Area),
- Plum District — One Reg Mgr opening in NYC/Manhattan and one in Brooklyn/Queens (Greater New York City Area).

We analyzed a huge amount of JobLocations parsed from crawled job offer pages (totally 70 116) and observed that in many cases there are multiple locations defined in one JobLocation. Due to uncertain location format and multiplicity of locations in one JobLocation string, it is not very smart to send the whole JobLocation string to the geocoding service and expect a successful result. There should be another approach used because multiple locations in one geocoding request do the job in confusing the geocoder to return erroneous results. There was a gazetteer approach considered for finding the sub-locations, but it was desisted from it because it would require a very precise gazetteer to cover as many as possible location names. Instead of it the location is split into several parts, where each part contains the possibly one sub-location.

**Table 3**
Index statistics for the LinkedIn task.

| | | | |
|---|---|---|---|
| Documents | | 113 268 | |
| Job offer pages | | 70 116 (61.90%) | |
| Terms | | 934 419 | |
| **Entity type** | **Docs** | **Doc. ratio** [%] | **Distinct** |
| JobLocation | 70 115 | *100.00 | 40 739 |
| JobLatLon | 69 992 | *99.82 | 12 796 |
| JobGeoHash | 69 992 | *99.82 | 12 792 |
| Address | 1038 | 0.92 | 582 |
| City | 101 602 | 89.70 | 6481 |
| Country | 40 623 | 35.86 | 224 |
| State | 62 020 | 54.76 | 97 |
| LatLon | 106 086 | 93.66 | 5011 |
| GeoHash | 106 086 | 93.66 | 5010 |
| Company | 27 265 | 24.07 | 11 084 |
| Experience | 70 115 | *100.00 | 10 |
| Industry | 70 115 | *100.00 | 186 |
| JobCompany | 70 100 | *99.98 | 19 042 |
| JobSkill | 70 115 | *100.00 | 51 186 |
| JobSkill2 | 67 967 | *96.94 | 32 702 |
| JobTitle | 70 115 | *100.00 | 51 186 |
| TelephoneNumber | 1324 | 1.17 | 864 |

During the JobLocation analysis, one can observe that sub-locations are often separated by conjunctions (e.g. “and”, “or”, “und”, “oder”, “y”, “o” in English, German, Spanish and other languages), which occur in geographic names very rarely. In addition, most of the JobLocation strings contain a “bracket part” describing wider geographic areas of the job. Both facts can be used to split one JobLocation string into several sub-location strings. Sub-location strings as the result of the split need to be further processed. Words, which are not typical for the geographic names and which occur quite a lot in the sub-location strings, like “anywhere”, “business”, “next”, “next to”, “office”, “work”, etc. are cleaned off. Afterwards, non-alphanumeric characters except the “.” and “&” are cleaned off as well. Finally, there are leading and trailing white-spaces trimmed.

After the cleaning and trimming process a set of sub-location strings and company names is almost ready for geocoding. But most of the sub-location strings cannot be yet precisely geocoded because they contain only city and country information (rarely, there is also a ZIP code). The geocoding service would return coordinates in the middle of the cities or countries, which is not sufficient. To get more precise geocoding results, one needs to make the location more specific, but this cannot be

done simply by specifying the company name in the query because geocoding services do not recognize business names.

But there is Google Places Autocomplete service [8], which can complete the address of some establishment, which we decided to use in the geocoding process. It takes establishment name, latitude, longitude and radius as input parameters. As the establishment, we put the company name and for latitude/longitude we put a geocoded sub-location string by the Google Geocoding service. Google Places Autocomplete service returns up to 5 results — complete addresses matched for the company near the specified latitude/longitude. Each result is then geocoded and its distance from the reference point is computed in order to filter distant results, which might be irrelevant. In addition a success probability for each geocoding service result is computed. The computation is based on the service return values "location_type" and "partial_match", which indicate the geocoding success. Then, latitude/longitude coordinates of the result with the highest probability are picked as a job location and stored in the index.

## 7. Conclusion

In this paper we have presented a work-in-progress framework for distributed crawling, extracting, indexing and lightweight semantic search over the extracted data with spatial support. The use of this framework has been shown on two example tasks, the LinkedIn job offer search task and the BBC news search task. Spatial indexing and searching has been implemented as a plugin for Nutch and Solr. This plugin has been used for indexing documents by more than one geographic location and for performing searches within a specified bounding-box (other options such as circle area can be easily implemented too).

Our future plans are to extend the semi-automated mapping between job offers and CVs (related to the LinkedIn task), to include job offers and CVs from the Monster website and to support other formats like PDF or DOC for the CV upload. Intelligent matching of job offers and users CVs to find the most suitable job for the applicant and the most suitable applicants for the job is our other goal. Last but not least, we want to invite users to use and evaluate the whole system from their point of view. Regarding the spatial index and search capabilities, we are working on their integration into Lucene since there is not yet multi-location indexing per document supported.

### Acknowledgements

*This work is supported by projects TRA-DICE APVV-0208-10, VENIS FP7-284984 and VEGA 2/0184/10. It is also the result of the projects implementation: SMART II ITMS: 26240120029 and ITMS: 26240220029 supported by Operational Programme Research & Development funded by the ERDF.*

## References


[1] Chang F., Dean J., Ghemawat S., Hsieh W. C., Wallach D. A., Burrows M., Chandra T., Fikes A., Gruber R. E.: Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26:4:1–4:26, June 2008.

[2] Ciglan M., Babik M., Šeleng M., Laclavik M., Hluchý L.: Running mapreduce type jobs in grid infrastructure. In *Cracow '08 Grid Workshop : proceedings*, 2009.

[3] Dean J., Ghemawat S.: Mapreduce: simplified data processing on large clusters. In *Proc. of the 6th conference on Symposium on Opearting Systems Design & Implementation* — vol. 6, pp. 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[4] Dlugolinsky S., Laclavik M., Hluchy L.: Towards a search system for the web exploiting spatial data of a web document. In *Proc. of the 2010 Workshops on Database and Expert Systems Applications*, DEXA '10, pp. 27–31, Washington, DC, USA, 2010. IEEE Computer Society.

[5] Dlugolinský Š., Laclavík M., Šeleng M.: Vyhľadávanie informácií na webe podľa vzdialenosti. In *Proc. of the 4th Workshop on Intelligent and Knowledge oriented Technologies*, WIKT 2009, Košice, Slovakia, November 2009. Equilibria.

[6] Gatial E., Balogh Z.: Identifying, retrieving and determining relevance of heterogenous internet resources. In P. Návrat *et al.*, ed., *Tools for Acquisition, Organisation and Presenting of Information and Knowledge, Research roject Workshop (NAZOU), in conjunction with ITAT 2006*, pp. 15–21, Bystrá dolina, Nízke Tatry, Slovakia, September 2006. Slovak University of Technology Bratislava.

[7] Google: The Google Geocoding API. `http://developers.google.com/maps/documentation/geocoding/`, May 2012.

[8] Google: The Google Places Autocomplete API (Experimental). `http://developers.google.com/maps/documentation/places/autocomplete`, May 2012.

[9] Habala O., Hluchý L., Tran V., Krammer P., Šeleng M.: Using advanced data mining and integration in environmental prediction scenarios. *Computer Science*, 13(1):5–16, 2012.

[10] Hahn R., Bizer C., Sahnwaldt C., Herta C., Robinson S., Bürgle M., Düwiger H., Scheel U.: Faceted wikipedia search. In W. Abramowicz, R. Tolksdorf, W. Aalst, J. Mylopoulos, M. Rosemann, M. J. Shaw, C. Szyperski, ed., *Business Information Systems*, vol. 47 of *Lecture Notes in Business Information Processing*, pp. 1–11. Springer, Berlin Heidelberg, 2010.

[11] Hearst M. A.: Design recommendations for hierarchical faceted search interfaces. In *SIGIR, Workshop on Faceted Search*, 2006.

[12] Kunszt P. Z., Szalay A. S., Thakar A. R.: The hierarchical triangular mesh. In A. J. Banday, S. Zaroubi, M. Bartelmann, ed., *Mining the Sky: Proc. of the MPA/ESO/MPE Workshop Held at Garching, Germany, July 31 – August 4,*

*2000*, volume XV of *ESO Astrophysics Symposia*, pp. 631–637, Springer-Verlag, Berlin Heidelberg, 2001.

[13] Laclavík M., Dlugolinský v., Šeleng M., Ciglan M., Hluchý L.: Emails as graph: relation discovery in email archive. In *Proc. eedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pp. 841–846, New York, NY, USA, 2012. ACM.

[14] Laclavík M., Šeleng M., Ciglan M., Hluchý L.: Supporting collaboration by large scale email analysis. In M. Bubak, M. Turala, K. Wiatr, ed., *Cracow'08 Grid Workshop*, pp. 382–387, Krakow, 2009. (Academic Computer Centre CYFRONET AGH)

[15] Laclavík M., Šeleng M., Hluchý L.: Towards large scale semantic annotation built on mapreduce architecture. In *Proc. of the 8th international conference on Computational Science, Part III*, ICCS '08, pp. 331–338, Springer-Verlag, Berlin, Heidelberg, 2008.

[16] LinkedIn Corporation: Apache Velocity template language. `http://www.linkedin.com/`, May 2012.

[17] Monster: Monster website. `http://www.monster.com/`, May 2012.

[18] Šeleng M.: Distribuované spracovanie dát nad mapreduce architektúrou (hadoop a hive). In *Proc. of the 5th Workshop on Intelligent and Knowledge oriented Technologies*, WIKT 2010, p. 141, Bratislava, Institute of Informatics SAS, November 2010.

[19] Slovak University of Technology in Bratislava, Institute of Informatics SAS, Pavol Jozef Šafárik University in Košice, Softec, Ltd.. NAZOU website. `http://nazou.fiit.stuba.sk`, May 2012.

[20] Szalay A., Gray J., Fekete G., Kunszt P. Z., Kukol P., Thakar A.: *Indexing the sphere with the hierarchical triangular mesh.* Technical Report MSR-TR-2005-123, Microsoft Research Advanced Technology Division, Microsoft Corporation One Microsoft Way Redmond, WA 98052, 2005.

[21] The Apache Software Foundation: Apache Lucene website. `http://lucene.apache.org`, May 2012.

[22] The Apache Software Foundation: Apache Solr indexing and searching framework. `http://lucene.apache.org/solr/`, May 2012.

[23] The Apache Software Foundation: Apache Velocity template language. `http://velocity.apache.org/`, May 2012.

[24] The Apache Software Foundation: Hadoop Distributed File System site. `http://hadoop.apache.org/hdfs/`, May 2012.

[25] The Apache Software Foundation: Hadoop site. `http://hadoop.apache.org/`, May 2012.

[26] The Apache Software Foundation: HBase site. `http://hbase.apache.org/`, May 2012.

[27] The Apache Software Foundation: Nutch site. `http://nutch.apache.org/`, May 2012.

[28] The Apache Software Foundation: Spatial search in Lucene.
`http://wiki.apache.org/lucene-java/SpatialSearch`, May 2012.

[29] W3Techs.: World wide web technology surveys.
`http://w3techs.com/`, October 2011.

[30] Yahoo! Inc.: Yahoo! PlaceFinder.
`http://developer.yahoo.com/geo/placefinder/`, May 2012.

## Affiliations

**Štefan Dlugolinský**
Institute of Informatics, Slovak Academy of Sciences, Bratislava, Slovakia,
`stefan.dlugolinsky@savba.sk`

**Martin Šeleng**
Institute of Informatics, Slovak Academy of Sciences, Bratislava, Slovakia,
`martin.seleng@savba.sk`

**Michal Laclavík**
Institute of Informatics, Slovak Academy of Sciences, Bratislava, Slovakia,
`michal.laclavik@savba.sk`

**Ladislav Hluchý**
Institute of Informatics, Slovak Academy of Sciences, Bratislava, Slovakia,
`ladislav.hluchy@savba.sk`

**Received**: 14.05.2012
**Revised**: 17.08.2012
**Accepted**: 3.09.2012