Cloud Based Processing Services Based on Linked Data

Elias Grinias[3(✉)] and Dimitris Kotzinos[1,2]

[1] ETIS Laboratory ENSEA UCP CNRS UMR 8501,
Department of Computer Science,
University of Cergy-Pontoise, Cergy-Pontoise, France
Dimitrios.Kotzinos@u-cergy.fr
[2] Institute of Computer Science,
Foundation for Research and Technology–Hellas (FORTH-ICS), Hellas, Greece
[3] Department of Civil Engineering, Surveying Engineering and Geoinformatics,
Technological Educational Institute of Central Macedonia, Serres, Greece
grinias@teiser.gr

Abstract. Cloud computing is providing a computing infrastructure to facilitate storage and processing of massive amounts of information (Big Data). Processing of massive datasets becomes more and more important since the data becoming available to us increase every day in volume, variety, speed of change and (potentially) quality. Processing these data becomes more and more difficult under traditional computing platforms since we need the ability to compute and scale at the same time.Under this context, for this work we describe the design and implementation of a responsive and user driven processing service. This is a geoprocessing service that operates on geospatial datasets and provides geostatistical interpolation (a specific variant called Kriging). This service is based on existing service implementation standards in the geospatial domain (namely WPS standard from OGC). Additionally our service can query and retrieve information that is integrated following the Linked Open Data (LOD) initiative. This is a unique capability that allows the service to rely on data, besides the existing ones and the ones provided by the user, that can be retrieved from the integrated information space that is being built on the web. In this paper we present the design and implementation of the service on a Linked Data store and discuss capabilities, issues and future research.

1 Introduction

Geoprocessing is considered a rather complex and computationally demanding processing activity regardless of the exact type of computation it involves; mainly we refer to statistical geoprocessing including various types of interpolation, that are of interest in this work. Given also the fact that geoprocessing refers to processing of geodatasets, which are usually of high volume, one can assume that this kind of processing is complex due to both computational complexity and data volume reasons.

D. Kotzinos et al. (Eds.): ISIP 2014, CCIS 497, pp. 97–112, 2016.
DOI: 10.1007/978-3-319-38901-1_7

On the other hand cloud computing promises scalable and elastic resources both for processing and storage of data. This has made many researchers to consider the cloud as one perfect match for solve complex geoprocessing problems that also need to be applied on large geospatial datasets. Many interesting works in this area exist already. One could reference the work of [17,18] and by [12] where a discussion takes place on how cloud computing can be used and shaped by the spatial sciences. Works like [16,19] shape the ground for a more in depth look into the algorithmic or technical needs of the geospatial cloud based applications. A very interesting comparison of various cloud based geospatial solutions can be found at [17]; although the authors are focused on Windows Azure and Google Application Engine platforms, the works compared are rather comprehensive and the conclusions can be extended to other cloud platforms like Amazon Web Services (AWS)[1], which is the one used for the work presented in this paper. In this respect our proposal is similar and complementary of those efforts, since (as we detail in Sects. 2, 3 and 4) we provide a standards based geoprocessing implementation and infrastructure.

At the same time an effort is underway to create more interconnected data sets; it has been evident that data published on the web cannot be fully exploited if they remain stored in information silos where no one but the owner will have access to. This effort claims better results if our data are created, published and re-used as Linked Data (LD), i.e., data that are inter-linked with each other and can be uniquely identified based on unique URIs. LD and the technology supporting them not only enables their re-use and interconnection but also allows for combining them on the fly, which adds value to the data and highlights and promotes their potential. In fact, nowadays, a great amount of LD is actually freely available and open on the web, thus leading to the Linked Open Data (LOD) concept. Such data are available for various areas either in raw RDF form or via SPARQL endpoints. The work presented on this paper provides geoprocessing facilities for Linked Open Data stored in an RDF triplestore in the cloud. Thus to the best of our knowledge is the only work, which actually retrieves and processes Linked Open GeoData by keeping both the data and the processing in the cloud.

The paper is organized as follows: Sect. 2 describes the preliminaries for the understanding of the Web Processing Service standards; Sect. 3 discusses the theoretical part of the specific geoprocessing algorithm used for statistical interpolation of values, called Kriging; Sect. 4 describes the implementation of Kriging/geoprocessing services according to the standards while Sect. 5 details the Linked Open Data (LOD) infrastructure and capabilities. In Sect. 6 the client that has been implemented to provide geoprocessing capabilities on Linked Open geodata is presented. The paper concludes with some conclusions and pointers for future work in Sect. 7.

[1] http://aws.amazon.com/.

2 Open Geospatial Consortium Web Processing Service

Web services are defined as software systems that allow the interaction between machines over a network. In such systems, there is often a machine-readable description of the operations offered by the service and the other systems communicate with the service using messages formatted in markup languages such as XML.

Web Processing Service (WPS) [7] is an Open Geospatial Consortium (OGC)[2] standard, which provides rules for standardizing the implementation of geographic calculations ("processes") as a web service. More specifically, the standard

- describes inputs and outputs (requests and responses) for invoking geospatial processing services, as a Web service,
- defines the way that a client can request the execution of a process, and how the output from the process is handled and
- defines an interface that facilitates the publishing of geospatial processes and clients discovery of and binding to those processes.

The Web Processing Service (WPS) standard defines three operations:

- **GetCapabilities** that returns metadata describing the service capabilities,
- **DescribeProcess** that returns a description of a process including its inputs and outputs and
- **Execute**, which returns the output(s) of a process.

In practice, WPS operations are invoked by submitting XML to the URL of the service. When requesting an Execute operation the HTTP request identifies the inputs, the name of process to be executed, and the form of output to be provided after execution. Data are often embedded in process execution input/output XML, although references to web-accessible data inputs/outputs are supported as well.

Input/output data required by the WPS can be delivered across the network or they can be available at the server. Three types of data are defined by the standard, namely:

- **Complex Data** such as imagery, XML, CSV, and custom (or proprietary) data structures,
- **Literal Data** for numerical values or strings and
- **Bounding Box** Data type for the geographic coordinates of a rectangular area.

3 Geoprocessing

3.1 Preliminaries

Kriging is a geostatistical method, which relies on the fact that as distance between points increases, the similarity, defined by the covariance or correlation

[2] http://www.opengeospatial.org/.

between points, decreases. Kriging predicts the unknown value $Z(\mathbf{x}_0)$ at a location in question $\mathbf{x}_0$ based on the data values in a neighborhood of this location. Similarly to other well known interpolation techniques, the calculation of the unknown value is based on a weighted sum of the locations with known values in the neighborhood of point $\mathbf{x}_0$:

$$\hat{Z}(\mathbf{x}_0) = \sum_{i=1}^{n} w_i(\mathbf{x}_0) Z(\mathbf{x}_i) \tag{1}$$

where weight $w_i(\mathbf{x}_0)$ is the contribution of value $Z(\mathbf{x}_i)$ and $n = N(\mathbf{x}_0)$ is the number of neighbors involved in predicting the unknown value. Unlike the deterministic interpolation methods, in Kriging the input data values are considered to be the realization $z(\mathbf{x})$ of a random field $Z(\mathbf{x})$ which consists of a trend $m(\mathbf{x})$ and a residual $R(\mathbf{x})$:

$$Z(\mathbf{x}) = m(\mathbf{x}) + R(\mathbf{x})$$

or

$$R(\mathbf{x}) = Z(\mathbf{x}) - m(\mathbf{x})$$

Kriging estimates the residual $R(\mathbf{x})$ as the weighted sum of the residuals at adjacent positions around the location point $\mathbf{x}$. Weights $w_i(\cdot)$ of Eq. (1) are derived from the covariance or the semivariance of known values and therefore semivariance modeling should statistically characterize the residual component.

The three basic variations of Kriging, namely Simple, Ordinary and Universal (or with trend), arise from the assumptions made about the trend component of input data as being known and constant (Simple), unknown and locally constant (Ordinary) and spatially or functionally varying (Universal Kriging), respectively. Both Simple and Ordinary techniques may be considered sub-cases of Universal Kriging. In addition, if the trend of Universal Kriging is not a function of spatial coordinates, then other known Kriging interpolation variants arise, such as Kriging with External Drift. Finally, if prediction refers to the average of the measured values in a particular area rather than to single points, we have the so-called Block Kriging.

3.2 Ordinary Kriging Method Analysis

Kriging interpolation consists of two steps, namely:

1. covariance, or semivariance modeling based on the set of locations with known values and
2. prediction of values for a number of points in question.

Semivariance Modeling. Kriging uses semivariance to express the degree of relationship between points on a surface. The empirical semivariance is half the variance of the differences between all possible points spaced a constant distance (lag) h apart:

$$\hat{\gamma}(h) = \frac{1}{2n(h)} \sum_{i=1}^{n(h)} (z(\mathbf{x}_i) - z(\mathbf{x}_i + \mathbf{h}))^2 \tag{2}$$

Semivariogram plots (empirical) semivariance values against lags h of distance. In practice, instead of the often noisy semivariance measurements which are obtained using Eq. (2) on the points with known values, a semivariance model, or function of the three parameters, *Range*, *Sill* and *Nugget* defined below, is used to compute the semivariance of point pairs according to their distance.

In theory, the semivariance value at the origin ($h = 0$) should be zero. If it is significantly different from zero for distances very close to zero, then this minimum semivariance value is referred to as the *Nugget* (Fig. 1). As points are compared to increasingly distant points, the semivariance increases. Beyond some distance, called *Range*, the values of any points on the surface are statistically uncorrelated. The semivariance value at $h = Range$ is called *Sill*.

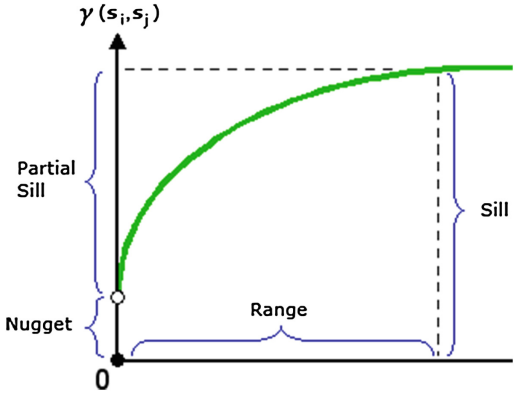


Fig. 1. Semivariogram and *Range*, *Nugget*, *Sill* (from ArcGIS Help 10.1: Semivariogram and covariance functions).

Prediction. Prediction may involve the overall set, or a subset of points with known values. In the first case we have global prediction. In the second case, a subset of points with known values is defined in an area of an acceptable, user-given radius (*SearchRadius*) around the point in question and only this subject is used for prediction (local neighborhood prediction). Furthermore, if the cardinality of this subset is less than a user-given value $MinNum$, no prediction is made ("bulls eyes" effect) and if exceeds a user-given value $MaxNum$, then only the $MaxNum$ points, which are closest to the point in question will be used in prediction. Furthermore, $MaxNum$ can be used on its own, without using search radius at all.

The steps taken to predict the unknown value at a specific location $\mathbf{x}_0$, given the set of points with known values are as follows:

1. First, distances between point $\mathbf{x}_0$ and each point with known value are computed.

2. Based on those distances, semivariance values between $\mathbf{x}_0$ and each one of the points with known values are computed, using the semivariance model.
3. Given the semivariance values, a series of linear equations is solved in order to get the predicted value for the location in question.

In case of using local neighborhood prediction, the steps above involve only the points that are placed in the local neighborhood of $\mathbf{x}_0$.

Furthermore, in most cases, interpolation refers to the prediction of values in locations of a grid that includes the points with known values. The parameters for grid construction, namely, grid extent in each dimension and grid cell size, may be given by the user or (grid extent for example), could be extracted automatically from the locations of points with known values.

3.3 Suitability for a Cloud Environment

As noted earlier, the cloud offers scalable unlimited (but not free) processing capabilities. From the discussion so far on Kriging prediction, one can notice that although the solving of linear equations is not complex, the execution time easily increases when a large set of points (or a large area in geospatial terms) is involved in the computation. In nowadays environments this can easily happen since we have both areas with very dense measurements (points) and large areas (e.g. Europe) where we need to perform interpolation computations.

As seen though in this work, we should also consider an additional factor for the increased demand: if this is a publicly available service we have literally no control on what kind of and how big datasets the users will upload in order to perform their calculations. Given also the fact that we might encounter situations where many concurrent users might want to use the service at the same time we could phase situations where a significant number of computations will take place at the same time but also on demand. These situations match perfectly the computational model of the cloud and thus make these services suitable for cloud based implementation.

4 Design and Implementation of a Geoprocessing Cloud Based Service

4.1 Open Source Kriging Implementations

In what follows, we refer to open source libraries or executable programs that provide Kriging interpolation implementations.

SAGA and SEXTANTE. The geospatial analysis library SAGA (System for Automated Geoscientific Analyses) [2] is implemented in C++ and includes processing modules for modeling the variograms as well as for performing Ordinary and Universal Kriging. The SEXTANTE (Sistema EXtremeño de ANálisis TErritorial) [13] library (coded in Java) includes the same functionality with SAGA, considering Kriging interpolation.

geoR. geoR [11] is a package of the open source, statistical processing environment R [10]. geoR includes modules for variogram modeling, as well as for applying Simple, Ordinary, Universal and external drift Kriging interpolation. The package is used by the v.Krige function of GRASS GIS [4], for applying Kriging techniques on input vector data.

HGPL. The HPGL (High Performance Geostatistics Library) library [5] (implemented in Python and C++) includes functions for variogram modeling and for applying simple, ordinary and generalized Kriging interpolation in the form of locally variable means. Data input as well as output results are stored in grids, as Eclipse Property or GSLIB [3] text files. Furthermore, the algorithms are applied on the Cartesian grid (IJK-grid) and the linear equations of Kriging techniques are solved using LAPACK solvers [1].

Gstat. Gstat [8] is a program dedicated to multivariable geostatistical modeling, prediction and simulation. It consists of a broad range of functionalities, which permit the efficient development of Kriging interpolation techniques. It was originally (1997) developed in ANSI C but, since 2003, its functionalities are available as an R package as well [9].

4.2 Servers and OGC-WPS Implementation

Among the many open source implementations of Kriging prediction available on the web, we selected the R [10] implementation of the Gstat [8] library (R-Gstat)[9] for performing Ordinary Kriging. Considering interpolation, R-Gstat supports

- Simple, Ordinary, Generalized as well as Block based Kriging prediction,
- global or local-neighborhood prediction,
- prediction on non-projected data using great circle distance between known points and
- fast enough prediction, since its main functionality is coded in C and local-neighborhood prediction is based on a fast neighborhood search algorithm.

WPS Ordinary Kriging process has been implemented using only open source software written in Java. The basic components of the overall system at server side (Fig. 2) are the Web Java Server and a WPS Java Container (implementation) installed in server's workspace, which provides the necessary functionality to handle responses to clients' requests for WPS processes' description/execution, according to the OGC-WPS standard. This way, developers are free to implement and publish web processes without having to worry about client/server interface and WPS processes' input/output issues.

Kriging process has been implemented as a Java class in a Linux machine, using Apache Tomcat[3] Web Java Server and the 52 North WPS[4] 3.1.1 implementation of OGC-WPS 1.0.0 standard [7]. Ordinary Kriging is applied on input data

[3] http://tomcat.apache.org/.
[4] http://52north.org/communities/geoprocessing/wps.

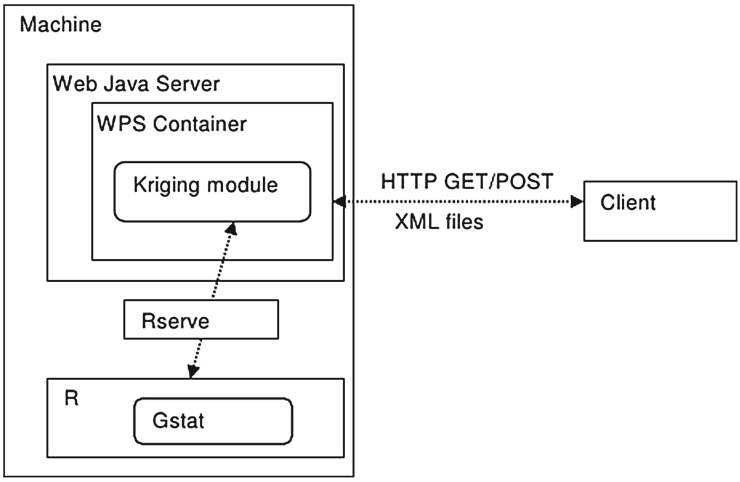


Fig. 2. Server configuration of WPS Kriging Implementation.

using R Gstat package. The interconnection between the Java module located at the WPS Container and R is handled by the TCP/IP server Rserve [15]. Rserve forwards to R the Java-R Interface (JRI) [14] instructions of the Java Kriging module and sends back to the module the returned output of each R instruction (if such an output exists), as it is depicted in the inner frame **Kriging Execution** of Fig. 5.

4.3 Geospatial Interpolation Process Implementation

The input of the process is handled by the WPS Container and consists of:

1. The input vector data (or layer) in the form of

$$[x, y, feature_1, feature_2, ..., feature_M]$$

   tuples, where $\mathbf{x} = (x, y)$ are the locations of vectors.
2. The field ($feature_j$) upon which Ordinary Kriging will be applied. It has to be a feature with arithmetic (real or integer) values.
3. The semivariance model that will be used. Corresponds to one of the variogram models supported by R Gstat.
4. The $Nugget$, $Sill$ and $Range$ values.
5. The $SearchRadius$ value, measured in kilometers for non-projected input data and in meters otherwise.
6. The $MinNum$ and $MaxNum$ values.
7. The cell size that will be used for constructing the grid with predicted values. Cell size should be given in meters for projected data and in degrees otherwise.

The Coordinate Reference System (CRS) of input vector data is assumed to be included in input data and if not, CRS EPSG:4326 (WGS84) is used by default. Grid extent is automatically computed by the extent of the corresponding input data.

The results of Kriging are three files, accessible as temporary links. The first file includes the kriging predictions in tab separated values (tsv) format, the second one is an image preview of Kriging predictions in PNG format and the third is a tsv file of the input data in $[x, y, feature_j]$ format. The first file includes Kriging predictions in the form $[x, y, predicted\_value, prediction\_variance]$ which (as it is depicted in the inner frame **Kriging Execution** of Fig. 5) after Gstat execution

1. are returned as R object in the opposite direction from R to the Java module through Rserve,
2. they are converted to Java arrays and
3. are written in the tsv file.

The second file is created by a plot function of R and is returned to the Java Server as well, using the file transfer capabilities of Rserve. The third file is constructed by the WPS at the Java Server side using the input data of the process. The OGC-WPS output XML (i.e. the response of **Execute** operation), which includes the three temporary links is then asynchronously returned to the client by the WPS Container.

5 Linked Open Geodata on the Cloud

There have been only limited efforts to publish Linked Open Geodata on the cloud. [6] provide a comparison among different efforts of publishing linked geodata on the cloud platforms and provide the description of an elastic and scalable service based infrastructure for providing Data-As-A-Service capabilities to any platform wishing to extend its application in Linked Data environments. The Linked Data Management API proposed in [6] carries very promising capabilities and allows for a seamless integration of the available Linked Data in various applications; its main architecture is depicted in Fig. 3.

One of the applications built on top of the LOD Management System is the Geoprocessing service, which has been described above, that retrieves data from the RDF Triplestore through the Linked Data API. Data are returned in RDF/XML format and then processed through the appropriate methods of the Geoprocessing Web Service. Querying the RDF Triplestore has been seamless and we had no actual trouble in retrieving the information in this format. Linked Data offer the opportunity to the geoprocessing module to combine data coming from different sources but refer to the specific area of interest. In that respect data coming from diverse sources can be easily integrated without the need of expensive (and most of the times incomplete) integration. The geoprocessing service will also use the Linked Data triplestore to store users own data that (s)he needs to upload in order to provide more input for better calculations. These data if described correctly using the appropriate ontology (-ies) can then

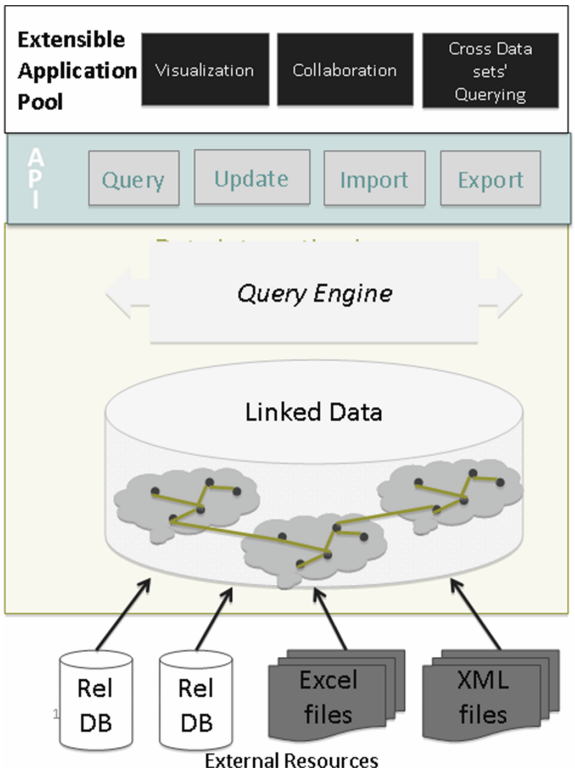


Fig. 3. The architecture of the LOD management system [6].

be linked with other data about the same area, allowing the scientists to draw better and more educated conclusions.

Up to its current implementation the service exploits the cloud only by allowing spawning of more instances in cases the load of the geoprocessing server becomes too big. Thus we use the standard AWS load balancer to account for high traffic or exceeding computational requirements (especially in cases of very complex statistical computations).

6 Client Application

For testing the WPS process and demonstrating its usage, a web client[5] has been implemented by modifying the open source 52 North Openlayers WPS Client[6]. In Fig. 5, the sequence diagram of client-server interaction is depicted in order to perform Kriging on Linked Open geodata using the web process implemented. In accordance with that Figure, the sequence of actions is as follows:

[5] http://portal.ingeoclouds.eu/sitools/client-user/Geoprocessing/project-index.html.
[6] https://wiki.52north.org/bin/view/Processing/52nOpenLayersWPSClient.

User Access. When the user navigates to the URL of client, (s)he sees the central html page built using styles and JavaScript (JS) libraries.

WPS Description. A panel has been developed as Openlayers control for giving the ability to the user to select input layer and Kriging parameters. To construct the panel, an HTTP GET request is send to the WPS container issuing the description of Kriging web process in terms of data inputs and parameters required for its execution (**DescribeProcess** operation). The response to that request is the OGC-WPS XML description file, which, after its asynchronous arrival at the client, is parsed in order to construct the panel (Fig. 4). WPS description step is executed upon loading of the main html page, and may be ignored in the case of a WPS with only one process. However, having available a mechanism of dynamically constructing the panel based on the DescribeProcess operation of OGC-WPS standard, leads to a highly extendable WPS client (e.g. in case of changing parameters of an already implemented process or of publishing new WPS processes).

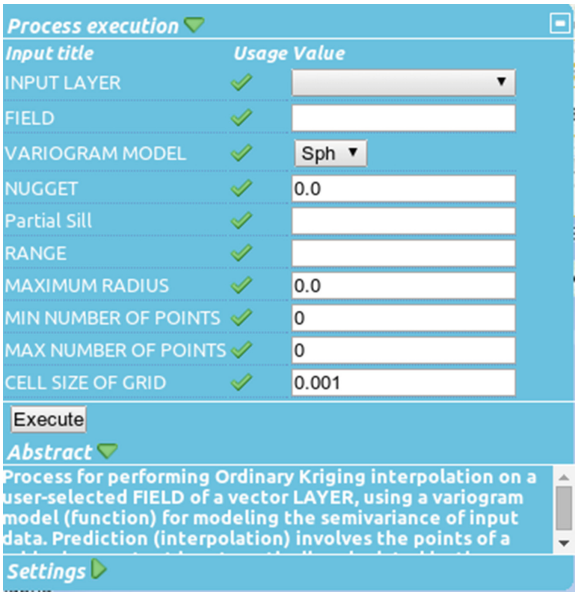


Fig. 4. User friendly Openlayers panel for giving Ordinary Kriging prediction input parameters.

Vector Layer Loading on Map. In current implementation, the user selects predefined queries, which are treated as "vector layers" of the Openlayers[7] library. Each time the user selects to load on the map a layer of that kind (using

[7] http://openlayers.org/.

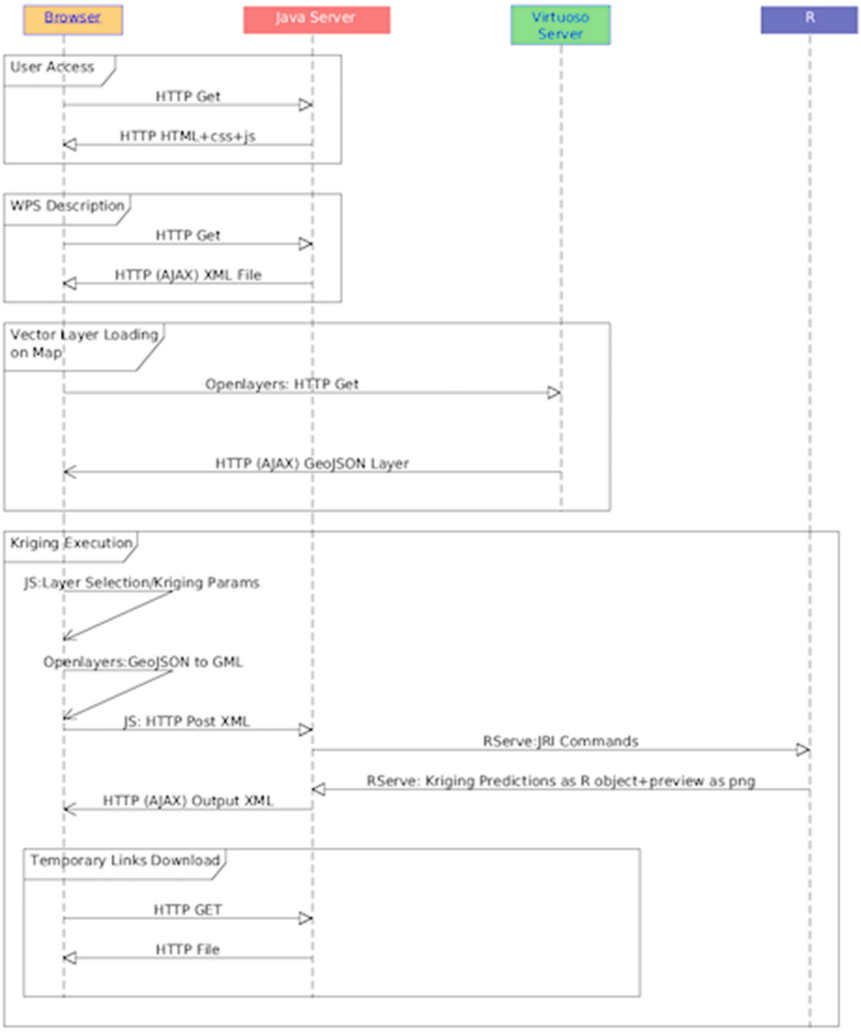


Fig. 5. Sequence diagram of client/server and server/R interconnections.

the JS control unit depicted in Fig. 6), a SPARQL[8] query is sent to Virtuoso[9] Server as HTTP POST request. The response is the result-set of the query which

1. is (asynchronously) sent back to Openlayers in GeoJSON[10] format,
2. is then transformed to an Openlayers layer and this layer is displayed on the Openlayers map and

[8] http://www.w3.org/TR/rdf-sparql-query/.
[9] http://virtuoso.openlinksw.com/.
[10] http://geojson.org/.

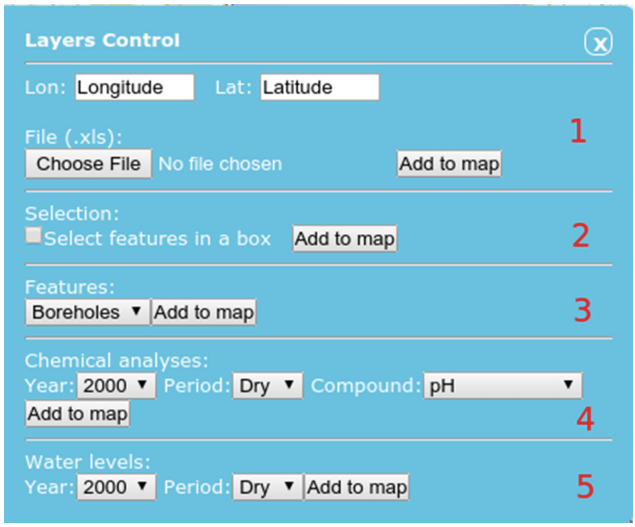
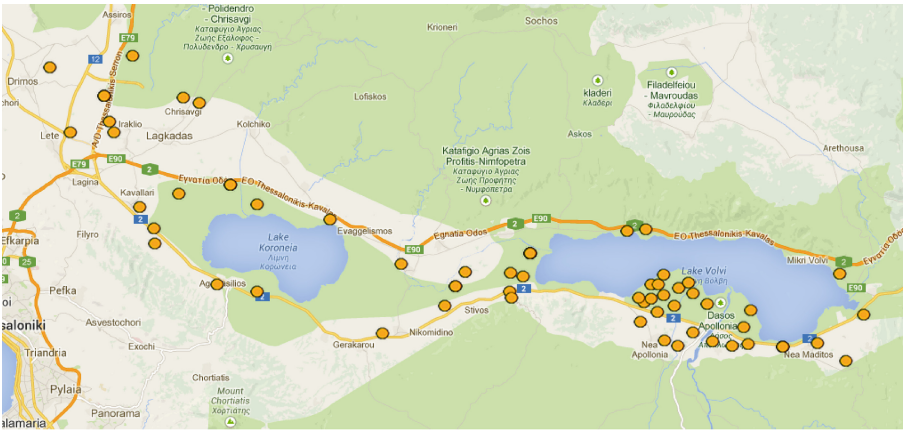


Fig. 6. Panel for loading vector layers on map. The user is given the ability to load (1) his own data in excel format, (3) linked geodata fetched from Virtuoso server using fixed SPARQL queries and (4),(5) linked geodata fetched from Virtuoso server using partially parameterized SPARQL queries. In addition, the user can create new vector layers using the rectangle box tool (2).

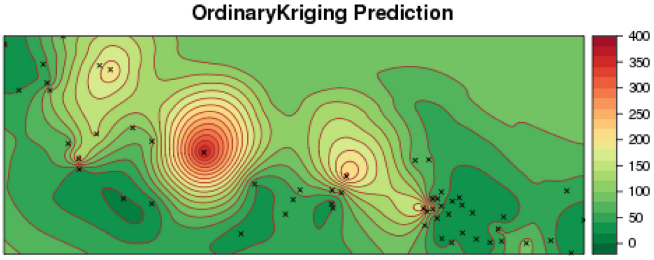
3. the corresponding entry of WPS panel with the layers that may be used as input data for Kriging is properly updated with the layer just loaded on the map.

Other sources of data could be used as well. Current implementation supports the loading on map of user data stored in Excel format. The procedure followed in that case is exactly the same: Excel data are first transformed to GeoJSON format and are then rendered on map as Openlayers vector layers. Furthermore, a tool has been implemented that permits the selection of features in a rectangle box. Using this tool the user can create new (Openlayers vector) layers from already loaded (on map) ones.

Kriging Execution. First, the user selects one of the loaded layers as the input data layer of the process and gives Kriging specific parameters. Then, the Openlayers layer is transformed to a format acceptable by 52 North OGC-WPS Java implementation (GML 2.0 in current implementation). The OGC-WPS input XML is then constructed, using the input data and Kriging parameters and is forwarded to the process through an HTTP POST request (**Execute** operation). The output XML is parsed using JavaScript and, after all, the user is given the ability by the user-interface to download the output files as described in the **Temporary Links Download** inner frame of the sequence diagram. In Fig. 7(b), the preview PNG image is shown, as it is returned by the WPS



(a)



(b)

Fig. 7. (a) Selected input layer (yellow points) and (b) preview of final interpolation result returned by the WPS process. Black crosses in (b) correspond to input data points. (Color figure online)

Ordinary Kriging interpolation process, applied on the selected input layer of Fig. 7(a).

7 Conclusions and Future Work

In this paper we introduced a Cloud based Processing Service that uses a Linked Open Data repository to retrieve its data and store user provided datasets. The service operates on a cloud environment and exploits the elasticity and scalability of the cloud mainly in the form of provision of more instances for processing when needed (scalability) and of storing the users datasets. The service uses Linked Open Data on the cloud, which is a unique feature of this work.

In the future we would like to explore techniques like Map Reduce that allow for distributed geoprocessing, something that in the case of Kriging for

example would considerably improve its performance. Additionally we would like to expand the service in order to provide geoprocessing on data that are automatically retrieved through their links on the web. Finally we would like to add more geoprocessing algorithms running in a cloud environment and run benchmarks to determine the differences in performance, scalability, elasticity and reliability between existing solutions and the cloud based one.

Acknowledgements. This work was partially supported by the INGEOCLOUDS project (Project reference: 297300) under the CIP-ICT-PSP.2011.4.1.

References

1. Anderson, E., Bai, Z., Bischof, C., Blackford, L.S., Demmel, J., Dongarra, J.J., Du Croz, J., Hammarling, S., Greenbaum, A., McKenney, A., Sorensen, D.: LAPACK Users' Guide. Society for Industrial and Applied Mathematics, 3rd edn. SIAM, Philadelphia (1999)
2. Böhner, J., McCloy, K.R., Strobl, J. (eds.): SAGA - Analysis and Modelling Applications, vol. 115. Verlag Erich Goltze GmbH, Göttingen (2006)
3. Deutsch, C.V., Journel, A.G.: GSLIB : Geostatistical Software Library and User's Guide. Oxford University Press, New York, Oxford (1992). http://opac.inria.fr/record=b1101614
4. GRASS Development Team: Geographic Resources Analysis Support System (GRASS GIS) Software. Open Source Geospatial Foundation (2012). http://grass.osgeo.org
5. HPGL Development Team: High Performance Geostatistics Library (HPGL) User Guide (2010). http://sourceforge.net/projects/hpgl
6. Kritikos, K., Rousakis, Y., Kotzinos, D.: Linked open geodata management in the cloud. In: Proceedings of the 2nd International Workshop on Open Data, WOD 2013, pp. 3: 1–3: 6 (2013)
7. OGC: OpenGIS Web Processing Service 1.0.0 (2007), openGISStandard, OGC05–007r7. http://www.opengeospatial.org/standards/wps
8. Pebesma, E., Wesseling, C.G.: Gstat: a program for geostatistical modelling, prediction and simulation. Comput. Geosci. **24**(1), 17–31 (1998). http://dx.org/10.1016/s0098-3004(97)00082-4
9. Pebesma, E.J.: Multivariable geostatistics in S: the gstat package. Comput. Geosci. **30**, 683–691 (2004)
10. Development Core Team, R.: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2011). ISBN3-900051-07-0. http://www.R-project.org/
11. Ribeiro Jr., P., Diggle, P.: geoR: a package for geostatistical analysis. R-NEWS **1**(2), 15–18 (2001). http://cran.R-project.org/doc/Rnews
12. Schäffer, B., Baranski, B., Foerster, T.: Towards spatial data infrastructures in the clouds. In: Painho, M., Santos, M.Y., Pundt, H. (eds.) Geospatial Thinking. LNGC, pp. 399–418. Springer, Berlin, Heidelberg (2010)
13. SEXTANTE development team: Sextante project official website (2011). http://www.sextantegis.com
14. Urbanek, S.: rJava: Low-level R to Java interface (2011). http://CRAN.R-project.org/package=rJava, r package version 0.9-3

15. Urbanek, S.: Rserve: Binary R server (2011). http://CRAN.R-project.org/package=Rserve, r package version 0.6-6
16. Wang, Y., Wang, S., Zhou, D.: Retrieving and indexing spatial data in the cloud computing environment. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) Cloud Computing. LNCS, vol. 5931, pp. 322–331. Springer, Heidelberg (2009)
17. Yang, C., Goodchild, M., Huang, Q., Nebert, D., Raskin, R., Xu, Y., Bambacus, M., Fay, D.: Spatial cloud computing: How can geospatial sciences use and help shape cloud computing. Int. J. Digit. Earth **4**(4), 305–329 (2011)
18. Yang, C., Raskin, R., Goodchild, M., Gahegan, M.: Geospatial cyberinfrastructure: Past, present and future. Comput. Environ. Urban Syst. **34**(4), 264–277 (2010)
19. Yue, P., Gong, J., Di, L., Yuan, J., Sun, L., Sun, Z., Wang, Q.: Geopw: Laying blocks for the geospatial processing web. Trans. GIS **14**(6), 755–772 (2010)