SOFTWARE Open Access

Hypermap registry: an open source, standards-based geospatial registry and search platform

Paolo Corti[1*], Benjamin Lewis[1] and Athanasios Tom Kralidis[2]

Abstract
On the web there is a large number of useful geospatial datasets available, exposed via web map services using open standards or open protocols. Just as web search engines enable users to reliably search and find relevant documents, a similar capability is needed to return the most useful and reliable geospatial datasets.
Hypermap Registry is an open source platform, developed by the Center for Geographic Analysis (CGA) of Harvard University, which attempts to address the general problem of geospatial data search and discovery.

Keywords: Catalogue, Data discovery, Geoportal, Interoperability, Metadata, OGC standards, Search engine, Spatial data infrastructure

Introduction

Thousands of web map services deployed using open standards and protocols, are being made available to the general public, which consumes a large volume of geospatial data across many disciplines and geographic areas. One major challenge is to provide a way to discover and use geospatial content from these services, which is accessible by the general end user [1].

In the past the problem of web service discovery has been addressed by primarily two approaches. The first approach uses a centralized registry, which is updated by the service providers [2]. The second approach uses crawling algorithms which harvest search engine application programming interfaces (APIs), such as the Google search API, in order to obtain standard web map services endpoints [1, 3, 4].

This research approaches the problem from a new perspective focusing on content relevance and reliability. Metadata matches and content returned to a client querying a registry should be: 1) relevant, using a ranked search score; 2) reliable, with respect to the web service availability. The registry should return quality content results to end users from web services which have a good reputation in respect of reliability.

In modern web sites, the problem of returning relevant content results to end users is addressed by using search engines frameworks. The open source information retrieval software library Apache Lucene [5] is currently used by a large number of web sites which require a powerful full-text searching mechanism. The same approach can be applied to return results from a registry of web map services.

Reliability of web map services can be measured by periodically querying remote map layers. Reliable web map services are those with a high percentage of correct responses to client requests. It is therefore important to have a reliability indicator, which would help to identify and optionally exclude low quality services from returned results [6].

Hypermap Registry is a free and open source platform, developed by the CGA of Harvard University, which tries to address this general problem: return relevant content from reliable web map services to users.

In this paper we first provide a description of the problem, which Hypermap aims to address. After a review of the underlying technologies we provide a background of the use cases for these technologies at Harvard CGA, including a description of the architecture and the features of the WorldMap platform and why we decided to design

*Correspondence: pcorti@fas.harvard.edu
[1]Center for Geographic Analysis, Harvard University, 1737 Cambridge Street, K350, 02138 Cambridge, MA, USA
Full list of author information is available at the end of the article

and develop the Hypermap platform. We then discuss the implementation and the architecture of Hypermap and provide an overview of its benefits. We conclude the paper with some plans we have for the future to extend our research and the developed platform.

Background

Harvard WorldMap is an open source Geospatial Content Management System (GeoCMS), containing a large number of geospatial datasets, which requires a framework to return to end users the most relevant and reliable results.

In this section we will provide an overview on GeoCMSs and their background problem domain and related technologies.

Geospatial content management systems

A Content Management System (CMS) is a web application, which allows users to work within a collaborative environment to create and update digital content. This content can exist in a number of different forms (blog posts, articles, images, videos etc) and it can be typically created, reviewed and published by applying a revision workflow and can be shared with other users and/or group of users through a granular permission system. In the last two decades a number of popular open source frameworks have gained popularity: some notable ones in the open source arena are Joomla, Drupal and Wordpress (based on the PHP language); Liferay (based on the Java language) and Plone and Django CMS (based on Python).

A different class of CMS, which we refer to as Data Content Management Systems (DCMS), is oriented toward the storage and distribution of open data and their metadata. CKAN is an open source web-based management system built on the Python/Pylons web framework, the PostgreSQL database, and Solr for search. CKAN is used by a number of public institutions to create data catalogues and registries, enabling users to share open data and make them discoverable and presentable. Data.gov, for example, is a United States government website built with CKAN, which has been in operation since 2009, and publishing open data at a national scale [7].

Another example for DCMS is Dataverse (built on Java/Glassfish), an open source web application that enables users to share, cite, explore and analyze research data. Its development started in 2006 at the Institute for Quantitative Social Science (IQSS) at Harvard University [8]. A growing number of universities and organizations around the world are running their own Dataverse repository instances.

Regarding the geospatial world, the focus of this paper, there are a number of frameworks being used to implement open geospatial web platforms. GeoCMS are DCMS which let the users create and update geospatial content and relative metadata. By using these frameworks it is possible to deploy Spatial Data Infrastructures (SDI) and/or geoportals.

A typical GeoCMS can provide users with some or all of the following abilities:

- Upload vector and raster datasets (layers), or create a vector dataset (layer) from scratch. Layers can be stored in a spatial database and rendered with a map rendering engine
- Create thematic styles for a layer
- Edit the metadata of a layer
- Edit the features of a vector layer
- Set appropriate granular permissions for a layer to a user or a group of users. Permissions can be of different types: for example a user can be enabled to access and edit the metadata of a vector dataset, but not enabled to edit its features and its styles
- Create web maps combining geospatial layers which have been uploaded to the GeoCMS
- Provide interoperability with external clients using standards and open protocols
- Harvest layers from external map services in order to make them available to the GeoCMS users

GeoCMS can be developed as a spatial extension of existing CMSs or as an independent framework which focus mainly on the management of spatial information. These systems are typically developed within a web framework that uses a spatial database to store the data and a map rendering engine to generate the map tiles.

Most of the open source GeoCMSs use PostgreSQL[1] with PostGIS[2] as spatial database and GeoServer[3] or MapServer[4] as the map rendering engine.

A JavaScript mapping library is used to implement the web maps, most common ones are OpenLayers[5] and Leaflet[6].

More sophisticated GeoCMS can include in the stack a map caching engine and a Catalogue Service for the Web (CSW)[7] instance.

One of the first GeoCMS frameworks was PrimaGIS (beginning of 2000), built on the Plone CMS using MapServer as a map rendering engine.

In the last decade the most widely adopted open source GeoCMS implementations have been GeoNode[8], Mapbender[9], Cartaro[10] and GeoNetwork (which evolved from a CSW implementation to a complete GeoCMS solution)[11]. CARTO (formerly CartoDB)[12] is a powerful cloud computing mapping platform. Its underlying source code, based on JavaScript, Node.js and PostgreSQL/PostGIS, is released as open source. There are however few CARTO instances running other than the main commercial one run by the CARTO company.

CKAN itself offers a number of geospatial features. For example, a non-spatial dataset can be geo-indexed and made it searchable by location by associating to it a GeoJSON geometry. Spatial search is performed by the spatial features of the CKAN search engine (Solr) or alternatively using a PostGIS spatial database. CKAN provides a way to harvest external CSW servers (federation). Thanks to the integration with pycsw, it can also provide a fully compliant CSW interface for the harvested records.

By using a GeoCMS, which uses a full stack of components (a spatial database, a map rendering engine, a map caching engine, a CSW catalogue), it is possible to implement an SDI and/or a solution for the storage and distribution of open data.

Search engines

CMS first, and later GeoCMS and open data portals started embedding a search engine in their architecture in order to make content and data easily discoverable. The most widely adopted open source search engine frameworks are Solr[13] and Elasticsearch[14], both of which are based on the Java Lucene[15] search library. Search engine technology provides fast scored results to end users and includes features similar to web search engines such as Google, Yahoo and Bing.

It is very common to pair a CMS with Solr or Elasticsearch. GeoCMSs have adopted this trend as well: in GeoNode, since the earliest versions, it has been possible to add Solr or Elasticsearch to the stack in order to make content more easily discoverable.

Web map services

On the web there is a large number of web map services exposing much useful geospatial information using Open Geospatial Consortium (OGC) standards or open protocols. Some popular OGC standards are Web Map Service (WMS) [9], Web Map Service Tile Caching (WMS-C)[16], Web Feature Service (WFS)[17], Web Coverage Service (WCS) [10], Catalogue Service for The Web (CSW). A set of very widely used and powerful open protocols are Esri ArcGIS Representational state transfer (REST) MapServer, ImageServer and FeatureServer[18].

Thanks to these standards and open protocols, a tremendous volume of geospatial information can be accessed by clients such as desktop platforms (QGIS, gvSIG, GRASS GIS, Esri ArcGIS, etc...) and by web platforms (GeoCMS and SDI) which can federate the services. For example GeoNode allows users to register a remote web map service in order to gain access to all of its published layers. These layers can then be used and combined with the native GeoNode layers to create web maps.

Harvard WorldMap

Since 2010 the CGA has been developing and maintaining WorldMap [11][19], a GeoCMS and open data platform that enables registered users to publish geospatial content on the web. Users can upload geospatial vector and raster datasets to the platform and combine them with existing datasets to create web maps. Existing datasets can be data which other users have uploaded, or data exposed by external web servers.

WorldMap is based on GeoNode and at the time of writing has been used by more than 20,000 registered users and provides access to about 120,000 map layers and 5000 maps (collections of layers).

In a WorldMap map object, users can combine local layers and remote layers (Fig. 1).

A local layer is a geospatial dataset managed by GeoNode: the data is stored as a PostgreSQL/PostGIS table if the layer is a vector dataset, or as a Geotiff file on disk if the layer is a raster dataset. In both cases the layer is displayed in the mapping client, which is based on the OpenLayers JavaScript library, using the OGC WMS standard [9] or the WMS-C specification, which are implemented by the GeoNode rendering engine, GeoServer. Whenever the client needs to access the coordinates of the feature's geometry, the OGC WFS [20] and the Web Feature Service - Transaction (WFS-T) standards, in GeoServer, are used.

A remote layer is a layer published in an external web map service. Most of the web map services are implemented using OGC Web Standards or specifications - like WMS, WMS-C, Tile Map Service (TMS), Web Map Tile Service (WMTS), WFS - or custom open protocols such as the Esri ArcGIS REST MapServer and ImageServer.

Hypermap registry

In a WorldMap map object it is possible to combine local GeoNode layers and remote layers from external web map services. Because there is a very large number of local and remote layers to search, a search platform is needed to enable WorldMap users to discover the most appropriate and reliable dataset for their specific need. The requirements of such a platform are:

- Enable creation and maintenance of a registry of web map services, exposed as OGC standards and Esri REST endpoints
- Make the collected geospatial information easily discoverable
- Constantly monitor layers status in order to filter out from users search layers which are not reliable: for example, layers which are published in not constantly up and running web map services
- Collect usage statistics to enable crowd curation of local and remote layers
- Provide instant previews (thumbnails) of local and remote layers
- Support visualization of geographic distribution of results returned
- Support robust search by time as well as space

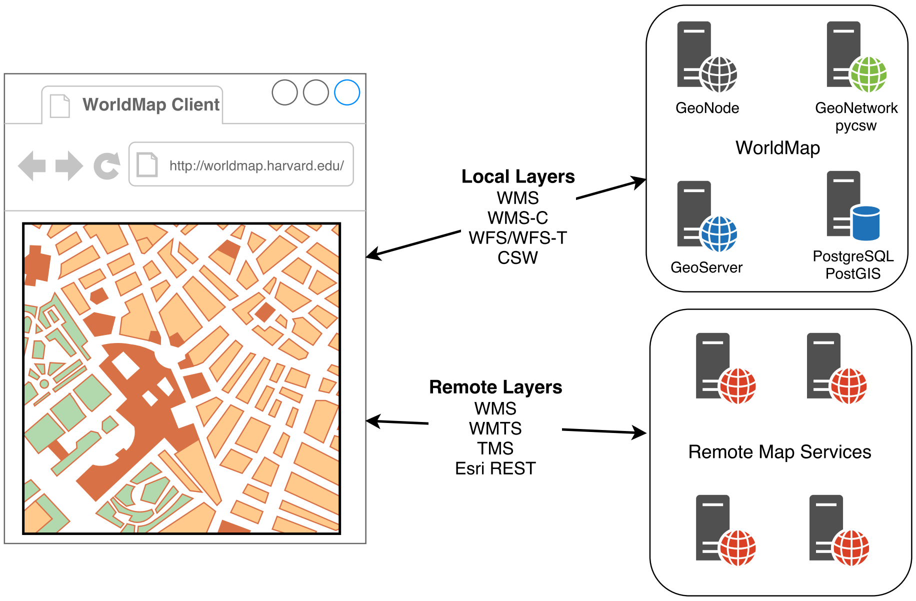


Fig. 1 Local and remote layers in WorldMap. Local layers are stored in GeoNode and loaded in the client using OGC standards implemented by GeoServer. Remote layers are exposed by a number of web map services using OGC standards or open protocols

In 2015 CGA started the design and development of **Hypermap Registry** (referred also as Hypermap), a registry platform to harvest and manage a large catalogue of web map services. Hypermap is released under the Massachusetts Institute of Technology (MIT) open source license, and is hosted on GitHub[21].

CGA runs a public instance of this platform, named **Harvard Hypermap (HHypermap)**[22], which is used by WorldMap to enable the users to search and use layers in their maps.

HHypermap implements a number of features in an attempt to provide high quality and reliable results to WorldMap users searching geospatial content published in web map services.

CGA staff has harvested a large number of web map services and layers using HHypermap. This information is handled in a relational database, and more services and layers can be added using the Hypermap user interface by system administrators or external users who can suggest map service endpoints. Hypermap is a web application with a public user interface providing users with access to the service information collected. Administrators can manage the information, for example adding a new map service to harvest or checking the health of known services.

Implementation

Conceptual model

Hypermap conceptual model is composed of three main entities: services, layers and checks (Fig. 2).

A **service** represents a remote map service, which can be implemented using an OGC standard (currently Hypermap supports WMS, WMTS, TMS) or an ArcGIS REST protocol (currently REST MapServer and ImageServer). OGC CSW, which provides a remote catalogue of layers and datasets, can also be used.

Each service publishes a capability document which exposes a number of layers. A layer is a geospatial dataset accessible on the web. For each layer published in a given service capabilities document, Hypermap harvests a layer thumbnail, based on a GetMap (or similar) web service request and information such as the name, the title, the

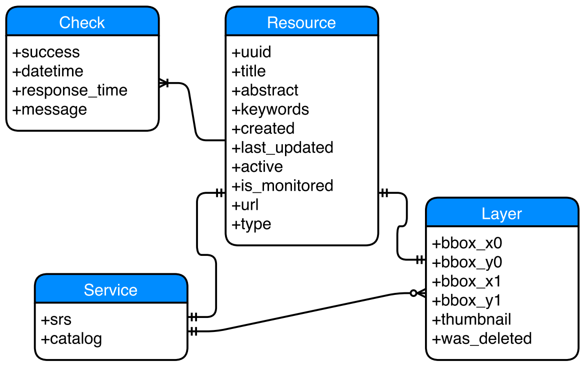


Fig. 2 Entity-relationship (ER) model of Hypermap. The main Hypermap entities are services, layers and checks. There is a 1-n cardinality between services and layers, service and checks and layers and checks

abstract, the keywords, the available spatial reference systems and its bounding box (Fig. 3).

A reality of remote services is variable service quality (downtime, slow performance, etc.). A remote service which is down for maintenance can be problematic for real-time applications. An important feature of Hypermap is the ability to periodically check the health of services and their layers, to provide end users with the most reliable ones. A **check** on a service is performed by sending an http request to the service capabilities document. If the request returns an http status code equals to 200 then the service check is considered successful. A check on a layer is performed by sending a GetMap (or similar) request to the layer. If an image is returned, then the check is considered successful, otherwise the check is unsuccessful and the returned error message is stored in the database.

For every check performed on a service or layer Hypermap collects high level metrics like request date/time, response time and status (success, failure). This way it is possible to generate a **reliability index**, which is the number of successful responses on the total number of http requests (Fig. 4). This index can be used by the Hypermap search engine to provide weighted results to the end users.

For each service and layer Hypermap stores temporal information about the **depict dates**. This information, when not provided by the layer's metadata, is automatically extracted by parsing textual metadata fields such as the title and the abstract [12].

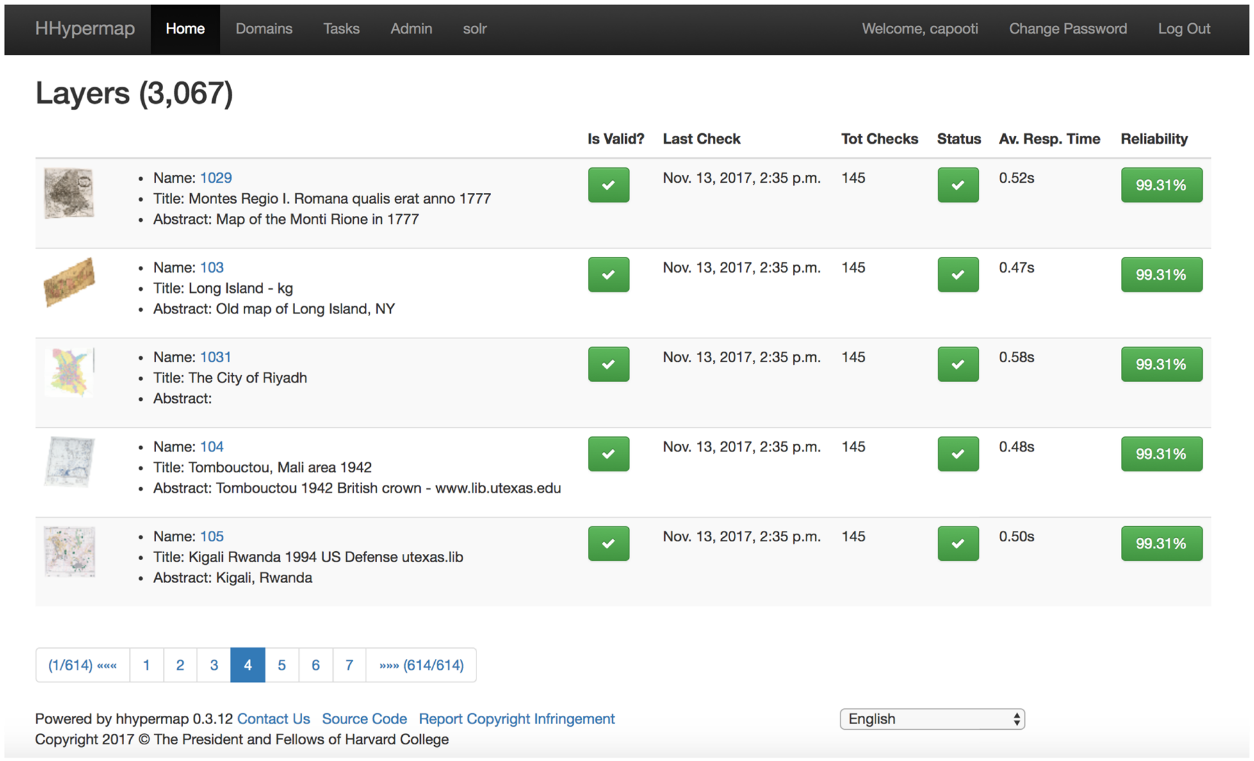
Hypermap provides a mechanism to periodically check the services and the layers. The registry needs to be updated frequently as new services and layers are published and older services and layers become stale. A system administrator is able to schedule the checking frequency for any service or group of services, or even for specific layers.

Architecture

Hypermap is a **web application** based on a **relational database** which is composed of a number of different components. The web application lets administrators manage the collected web map services and their layers, and their corresponding checks.

The system implements an **OGC CSW endpoint**, which enables external systems to search the Hypermap catalogue metadata using the CSW standard to provide broad interoperability to both mass market applications as well as geospatial professionals or subject matter experts.

In order to provide an enhanced search experience to end users and return results as fast as possible, a **search engine** has also been added to the stack. Therefore Hypermap needs to synchronize part of the information



Layers (3,067)

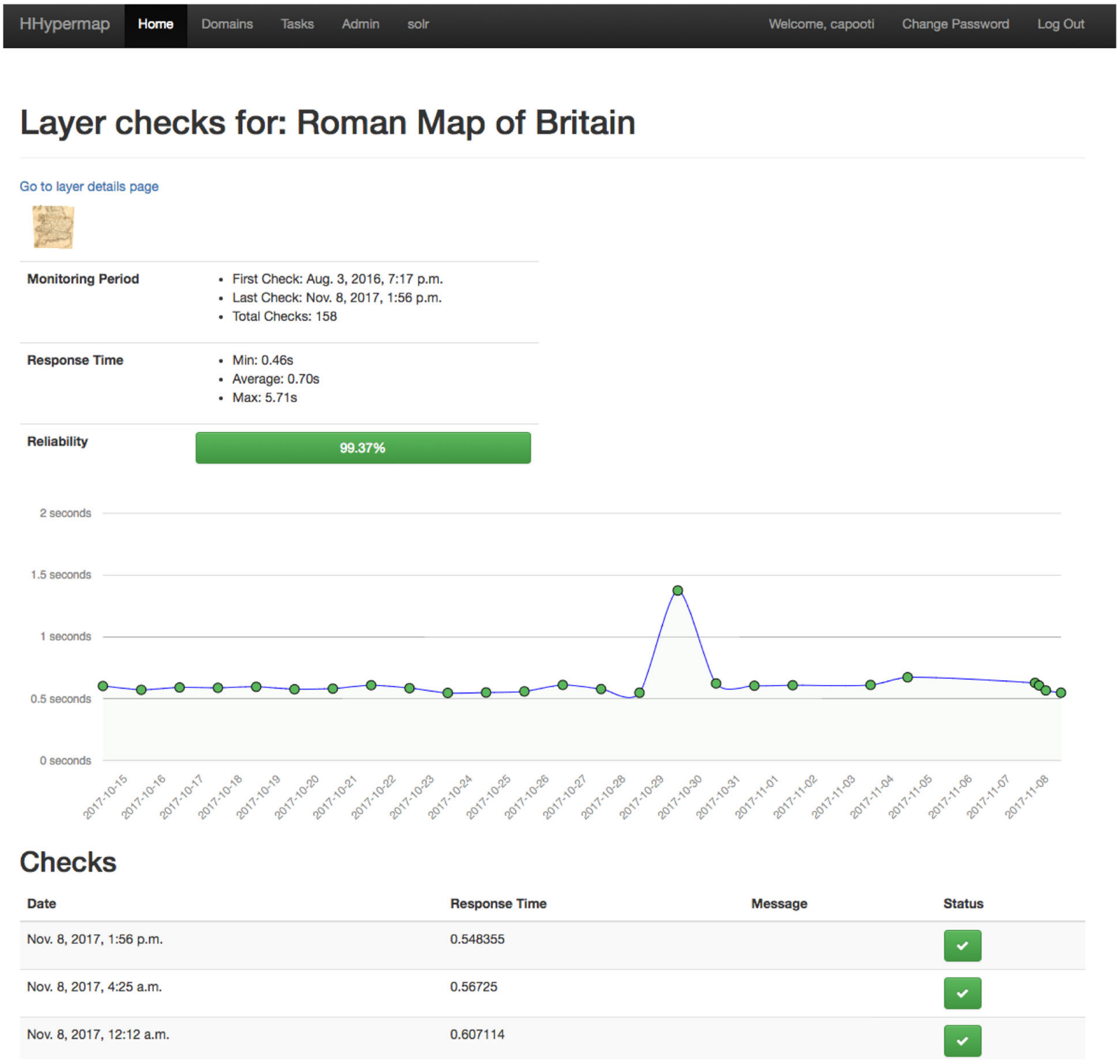
		Is Valid?	Last Check	Tot Checks	Status	Av. Resp. Time	Reliability
	• Name: 1029 • Title: Montes Regio I. Romana qualis erat anno 1777 • Abstract: Map of the Monti Rione in 1777	✓	Nov. 13, 2017, 2:35 p.m.	145	✓	0.52s	99.31%
	• Name: 103 • Title: Long Island - kg • Abstract: Old map of Long Island, NY	✓	Nov. 13, 2017, 2:35 p.m.	145	✓	0.47s	99.31%
	• Name: 1031 • Title: The City of Riyadh • Abstract:	✓	Nov. 13, 2017, 2:35 p.m.	145	✓	0.58s	99.31%
	• Name: 104 • Title: Tombouctou, Mali area 1942 • Abstract: Tombouctou 1942 British crown - www.lib.utexas.edu	✓	Nov. 13, 2017, 2:35 p.m.	145	✓	0.48s	99.31%
	• Name: 105 • Title: Kigali Rwanda 1994 US Defense utexas.lib • Abstract: Kigali, Rwanda	✓	Nov. 13, 2017, 2:35 p.m.	145	✓	0.50s	99.31%

(1/614) ««« 1 2 3 4 5 6 7 »»» (614/614)

Powered by hhypermap 0.3.12 Contact Us Source Code Report Copyright Infringement
Copyright 2017 © The President and Fellows of Harvard College

English

Fig. 3 Service details page in Hypermap. The service details page shows the main information for a given service and its exposed layers



Layer checks for: Roman Map of Britain

Monitoring Period	• First Check: Aug. 3, 2016, 7:17 p.m. • Last Check: Nov. 8, 2017, 1:56 p.m. • Total Checks: 158
Response Time	• Min: 0.46s • Average: 0.70s • Max: 5.71s
Reliability	99.37%

Checks

Date	Response Time	Message	Status
Nov. 8, 2017, 1:56 p.m.	0.548355		✓
Nov. 8, 2017, 4:25 a.m.	0.56725		✓
Nov. 8, 2017, 12:12 a.m.	0.607114		✓

Fig. 4 Layer checks page in Hypermap. The layer checks page shows statistics about checks performed in the time for a given layer

stored in the relational database to the search engine indexed content.

Hypermap provides a **RESTful Application Programming Interface (API)** that can be used to query the search engine which abstracts the search engine technology being used (Solr or Elasticsearch).

Hypermap cannot process all of the user requests synchronously because some require more time than is ideal for the http request response cycle. A **task queue** is needed to process asynchronously the tasks generated by these requests. Typical tasks are the check of a service or layer and the synchronization of the layer information between the relational database and the search engine.

Hypermap also includes a **map cache engine**, which caches and re-projects layers from the web map services on the fly. Caching a given layer's tile greatly accelerates the response time for clients, assuming the tile had been previously generated from the remote service - typically by a previous request or with a layer pre-seed process. Another benefit of using a map cache engine is to support a layer re-projection operation to a different spatial reference system which is not supported by the original remote service.

In the following sections we provide a description of the implementation for each of the different components of Hypermap: web application, relational database, search engine, application API, map cache, OGC Catalogue, task queue (Fig. 5).

Web application

Hypermap is a web application based on **Django**[23], a Python web framework well suited for handling information structured in a relational database.

The application uses a number of external Python libraries, most notably:

- OWSLib, which enables client programming with OGC Web Services (OWS) supported by Hypermap: CSW, WMS, WMTS, TMS
- arcrest, which enables client programming with Esri ArcGIS REST services
- psycopg2, a PostgreSQL adapter for Python
- Celery, which together with RabbitMQ, an open source messaging broker, implements a distributed task queue
- djmp, which enables the use of MapProxy within the Django environment
- pysolr and elasticsearch, which enable client programming with Solr and Elasticsearch

The application leverages design patterns for service checking and metrics from GeoHealthCheck, a service status and quality of service checker for OGC Web Services[24].

Relational database

Hypermap Relational Database Management System (RDBMS) is based on **PostgreSQL**, an advanced open source database. In Hypermap, all of the needed information is managed using different relational tables. The main tables contain the information related to the Hypermap entities, the main ones being services, layers and checks. Other tables are used by Django to store the metadata of the web application, and others are used by the task queue component to schedule tasks and their results.

Search engine

The approach of combining search engine with a traditional web portal in order to enhance the search user experience can be extended to SDI and GeoCMS [13]. The Hypermap search engine is based on **Solr** or **Elasticsearch**. Both are based on the Apache Lucene text search library. The search engine enriches the search user experience by providing:

- Extremely fast and scalable search, thanks to the indexed structure of the content
- Support for results faceting, providing search results in categories based on indexed terms. For example metadata results can be arranged by topic category, keyword and region
- Spatial facets provide a spatial surface representing the geographic distribution of layers
- Temporal facets can be used to arrange results by temporal histogram
- Ability to handle the ambiguities of natural languages, thanks to stopwords (words filtered out during the processing of text), stemming (ability to detect words derived from a common root),

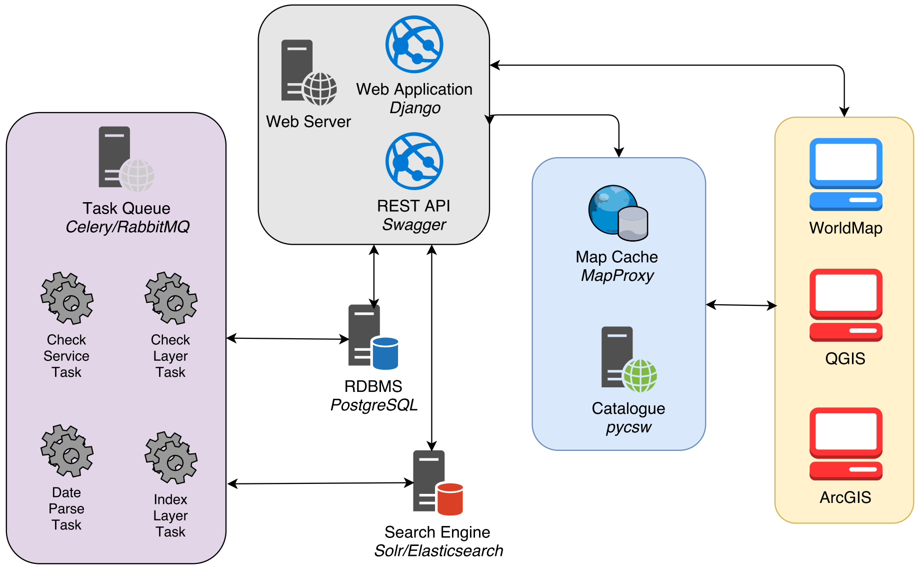


Fig. 5 Architecture of Hypermap. Hypermap is composed of a web application and API, a relational database, a search engine, a map cache, an OGC catalogue and a task queue. The mapping clients interact with the Hypermap components using standards over http requests

synonyms detection, and controlled vocabularies such as thesauri and taxonomies

- Scored results, providing more relevant results at the top
- Support for regular expressions, wild-card, and fuzzy search capabilities
- Support for boolean queries
- Hit highlighting, which provides immediate search term suggestions

Application API

The Hypermap application API provides a convenient way to query the search engine content using REST. The API implementation is based on **Swagger**[25], which is a framework of API developer tools for the OpenAPI Specification. Swagger provides a convenient way to clearly render API definitions, in a way that is search engine agnostic. This way it is possible to query the API in the same manner whatever is the search engine technology being used (Solr or Elasticsearch).

The Hypermap application API is used by the WorldMap search interface, from where users are able to search layers by keyword, source, layer type, bounding box and temporal range.

Map cache

The Hypermap map cache component (Fig. 6) is based on **MapProxy**[26], an open source proxy for geospatial data written in Python, which caches, accelerates (up to 100 times for some WMS) and re-projects layers from existing map services. MapProxy is based on configuration files: Hypermap generates a configuration for each layer which can be used by external clients, like WorldMap, to retrieve maps using standards like WMS-C, TMS, WMTS.

The MapProxy configuration file and a demo of the cached layer are accessible from the Hypermap layer details page.

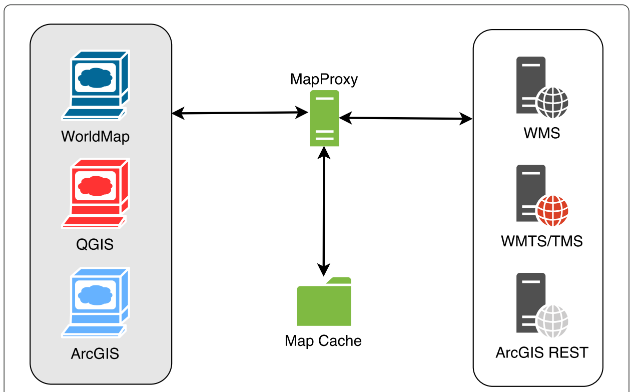


Fig. 6 Architecture of Hypermap map cache component. The map cache component, MapProxy, acts as a proxy to the external map services and provides a quick access tiles cache with support to the spatial reference systems used in the WorldMap client

OGC catalogue

Hypermap OGC Catalogue component is based on **pycsw**[27], which is an OGC CSW server implementation written in Python. Because of the pycsw entry point, users are able to query Hypermap metadata using the CSW standard, allowing for broad interoperability across multiple platforms and decision support systems.

For example it is possible to use the MetaSearch plugin in the popular QGIS desktop GIS software or the CSW Client in Esri ArcGIS products to search Hypermap and add remote map layers discovered to the map canvas.

pycsw uses the Hypermap RDBMS directly as its metadata repository. This is made possible by a custom pycsw plugin for Hypermap developed for this purpose.

Task queue

The task queue component is based on **Celery/ RabbitMQ**[28]. Hypermap requires a task queue to process large number of asynchronous tasks such as:

- Layer check task: a layer check is performed with a GetMap (or equivalent) request to a WMS (or equivalent) service type. This kind of requests can take up to some seconds to be processed, so it is always better to process them asynchronously from the regular http request/response cycle
- Service check task: a service check is performed with a GetCapabilities (or equivalent) request to WMS (or equivalent) service type. There are cases with very large (or slow, or both) GetCapabilities documents, which are therefore time consuming and need to be processed asynchronously. A service check typically generates a child layer check task for each one of the service layers
- Layer synchronization task: every time a layer is checked, the layer information is updated by Hypermap in the relational database. But part of this information must be synced to the search engine with an http request, which is better processed asynchronously in the task queue

In Hypermap there can be thousands of tasks which need to be run in the queue daily. The task queue implementation provides a convenient way to scale when task numbers increase, by adding more parallel workers to the execution stack. The task queue also provides a handy way to run periodic tasks, for example services which need to be checked regularly, and the ability to inspect the status of processed tasks.

Results and discussion

The CGA developed Hypermap and maintains its HHypermap instance with one aim being to provide to WorldMap users an enhanced search experience with quality and reliable results. The broader aim is to start to address the general problem of search for map service layers outside of WorldMap by creating an open source platform for building and maintaining large registries which can be used by a variety of clients. We consider the WorldMap search client to be a prototype for the development of others which run in other systems and provide their users with powerful geo-search capabilities.

When creating a map, the WorldMap user can upload new layers and combine them with the thousands of local and remote layers indexed in HHypermap. The user is able to search the layers by keyword, source, layer type, map extent, and date range (Fig. 7).

The WorldMap search interface queries the HHypermap search engine via the application API, returning the most relevant results in a tabular view as well as a spatial view.

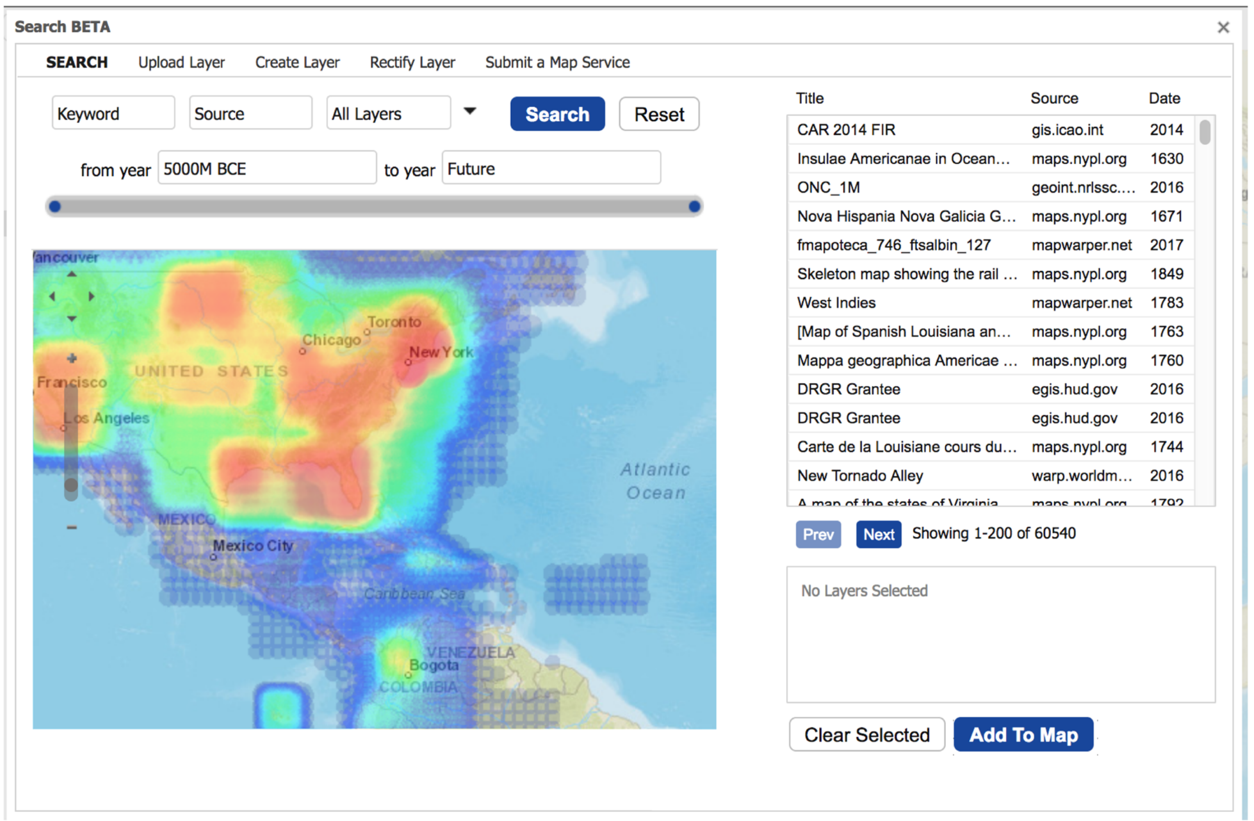
The tabular view is built based on the JavaScript Object Notation (JSON) results returned by a query to the search engine. The query includes all of the parameters which the user set in the WorldMap search interface.

The spatial view displays a heat map around the number of layers found on each cell of a spatial regular grid, which is generated from the results returned by the spatial facets feature of the search engine.

A number of advantages for searching content have been introduced in WorldMap by the use of the Hypermap API. Queries to the search engine are extremely fast as the metadata content has been pre-indexed during the tasks which synchronize the content between the relational database and the search engine itself. Results from a search engine are returned to the client much faster than from a CSW catalogue implementation based on a relational database. Thanks to a distributed architecture, the search engine is also extremely scalable.

Furthermore, when compared to a CSW catalogue, which searches metadata using a full text approach, the search engine is able to detect a larger number of matching metadata documents, as it is able to use thesaurus, text stemming and synonyms detection to recognize common variations of a given keyword or sentence.

Table 1 presents the evidence that a search engine outputs more results and faster than a CSW catalogue backed by a RDBMS. We selected a number of very common keywords appearing in the HHypermap layers metadata (a total number of about 120,000 documents, one for each layer): for each of them we calculated how many matching metadata documents are returned respectively from the WorldMap search engine, based on Solr, and from



Keyword | Source | All Layers

from year 5000M BCE to year Future

Title	Source	Date
CAR 2014 FIR	gis.icao.int	2014
Insulae Americanae in Ocean…	maps.nypl.org	1630
ONC_1M	geoint.nrlssc.…	2016
Nova Hispania Nova Galicia G…	maps.nypl.org	1671
fmapoteca_746_ftsalbin_127	mapwarper.net	2017
Skeleton map showing the rail …	maps.nypl.org	1849
West Indies	mapwarper.net	1783
[Map of Spanish Louisiana an…	maps.nypl.org	1763
Mappa geographica Americae …	maps.nypl.org	1760
DRGR Grantee	egis.hud.gov	2016
DRGR Grantee	egis.hud.gov	2016
Carte de la Louisiane cours du…	maps.nypl.org	1744
New Tornado Alley	warp.worldm…	2016
A map of the states of Virginia	maps.nypl.org	1792

Showing 1-200 of 60540

No Layers Selected

Fig. 7 WorldMap search user interface. The WorldMap search user interface let the user to query the Hypermap search engine using the Hypermap API

Table 1 The search engine outputs a larger number of matches and with a faster response time when compared it to a CSW catalogue based on a relational database

	Search engine		CSW catalogue	
Keyword	Number of results	Average response time (ms)	Number of results	Average response time (ms)
Agriculture	879	0.015	444	1.297
Climate	1289	0.017	783	1.317
Farming	522	0.016	50	1.327
Health	1338	0.016	779	1.335
Ocean	767	0.016	476	1.344
Planning	2080	0.015	760	1.297
Population	1471	0.018	1302	1.317
Society	3162	0.016	157	1.328
Transportation	2197	0.017	1091	1.362
Utilities	1082	0.016	350	1.308

the WorldMap CSW catalogue, based on the PostgreSQL RDBMS. The average response times (in ms), calculated on 100 different queries, are also provided. Both Solr and PostgreSQL are using a default installation and configuration in a Linux Ubuntu environment. To generate the table we developed a Python script with the owslib and pysolr libraries to interact with pycsw and Solr[29].

As it can be noticed, returned results from the search engine can be up to 20 times the number of results returned from the CSW catalogue based on a relational database, as in the case of the keyword "society". This is because the catalogue uses the full text feature of the relational database, while the search engine, with its powerful feature set, returns metadata containing all of the common variations and synonyms for a given keyword.

At the time of writing, in the HHypermap database CGA administrators have harvested about 15,000 services for a total of about 120,000 layers. The services and layers are regularly checked and re-indexed in the search engine as needed. This is possible by creating periodic operations from the Hypermap administrative interface to perform service and layer checks and for service and layer synchronizations between the RDBMS and the search engine. These periodic operations are scheduled and executed in the task queue.

HHypermap provides a reliability index for each layer and service, which is computed as the number of successful checks on the total number of checks. It is interesting to note that there are a number of services and layers which have a low reliability index. It is possible to disable these services or layers to prevent them from being returned or to weight them such that they disappear from search results.

Table 2 presents the first five layer metadata documents returned by Hypermap search engine for the search term "agriculture". For each document in the table there is a row which displays the ID of the service to which the layer belongs to, the layer title, the number of checks performed on the layer, the last check date, the reliability of the layer and the search engine score. The search engine score is based on the Lucene scoring model, which involves a number of scoring factors, most notably the term frequency and the proportion between the search term and the total length of the searched fields [5].

As it can be deducted from the table, a layer with a high search engine score can have a low reliability, which is not desirable for end users. In the same way, it is possible to have very reliable layers but with a low search engine score. While at the time of writing Hypermap uses only search engine score to order results, in future versions it will combine reliability with the search engine score to

Table 2 Results from Hypermap API for a keyword search with the term "agriculture"

Service ID	Layer title	Total checks	Last check date	Reliability (%)	Search score
23	Makkah 1947 Agriculture	13	02/02/2018	92.308	9.139
12006	Gross Value Add (2008)	5	02/19/2018	80.000	8.950
4663	Agriculture Regions	5	02/19/2018	100.000	8.855
1028	Agriculture Census	4	02/19/2018	100.000	8.855
23	Urban Agriculture	13	02/02/2018	84.615	8.855

return to the users the most appropriate layers. Another criterium which we are considering to better define results is to use layer usage statistics: Hypermap layers which are used in a large number of WorldMap maps will have a higher score.

An additional feature provided by Hypermap is that layers are cached by the Hypermap map cache, based on MapProxy, making remote map layer display fast and reliable. Significantly, this enables instant previews of any given layer during the search process. Even if a map layer tile is not available at the time it is requested, Hypermap can still return the cached tile from MapProxy, provided it has been already generated from a previous request or in case the layer has been pre-seeded when it was accessible.

Conclusions

Hypermap is an open source software platform which enable organizations to create large catalogues of reliable and discoverable map layers.

While the main aim of HHypermap is to provide fast and reliable search results to WorldMap users, Hypermap can be used by organizations in a variety of scenarios including:

- To monitor and health check the layers of a given SDI
- As a CSW catalogue of the SDI itself providing both registry and catalog functionality that includes service monitoring
- To enhance the catalogue of an SDI with the speed and features of a search engine
- To implement in a geoportal a search experience based on keywords, as well as temporal and spatial faceting
- As a user interface to MapProxy, to proxy local and remote layers to make them fast, more reliable and to reproject them as needed
- As a complement to a running GeoNode instance in order to provide support for remote services and layers

The authors hope to eventually develop additional features:

- A search engine backend for pycsw to improve the speed and quality of results returned from catalogues
- Support for the WFS standard and Esri Feature Service protocol to enable user search at the feature level in addition to the layer level
- Include more web service types in the supported service type list

The authors also plan to implement Hypermap Registry as a native application for GeoNode, for the management of remote services. Currently the GeoNode remote services application is limited, as it doesn't provide the search engine integration and the monitoring tools which are available in Hypermap.

Availability and requirements

- Project name: Hypermap Registry
- Project home page: https://github.com/cga-harvard/Hypermap-Registry
- Archived version: https://doi.org/10.5281/zenodo.1044763
- Operating system(s): Platform independent
- Programming language: Python
- Other requirements: nginx or httpd, uwsgi, PostgreSQL, Solr/Elasticsearch, RabbitMQ
- License: MIT

Endnotes

[1] PostgreSQL. https://www.postgresql.org/. Accessed 14 April 2018

[2] PostGIS. http://postgis.net/. Accessed 14 April 2018

[3] GeoServer. http://geoserver.org/. Accessed 14 April 2018

[4] MapServer. http://mapserver.org/. Accessed 14 April 2018

[5] OpenLayers. https://openlayers.org/. Accessed 14 April 2018

[6] Leaflet. http://leafletjs.com/. Accessed 14 April 2018

[7] OGC CSW. http://www.opengeospatial.org/standards/cat. Accessed 14 April 2018

[8] GeoNode. http://geonode.org/. Accessed 14 April 2018

[9] Mapbender. https://mapbender.org/. Accessed 14 April 2018

[10] Cartaro. http://cartaro.org/. Accessed 14 April 2018

[11] GeoNetwork. https://geonetwork-opensource.org/. Accessed 14 April 2018

[12] Carto. https://carto.com/. Accessed 14 April 2018

[13] Apache Solr. http://lucene.apache.org/solr/. Accessed 14 April 2018

[14] Elasticsearch. https://www.elastic.co/. Accessed 14 April 2018

[15] Apache Lucene. https://lucene.apache.org/. Accessed 14 April 2018

[16] WMS-C. https://wiki.osgeo.org/wiki/WMS_Tile_Caching. Accessed 14 April 2018

[17] OGC WFS. http://www.opengeospatial.org/standards/wfs. Accessed 14 April 2018

[18] Esri ArcGIS REST API. https://developers.arcgis.com/rest/. Accessed 14 April 2018

[19] WorldMap web site. http://worldmap.harvard.edu/. Accessed 14 April 2018

[20] OGC WFS. http://www.opengeospatial.org/standards/wfs . Accessed 14 April 2018

[21] Hypermap. https://github.com/cga-harvard/Hypermap-Registry. Accessed 14 April 2018

[22] Harvard Hypermap. http://hh.worldmap.harvard.edu/. Accessed 14 April 2018

[23] Django. https://www.djangoproject.com/. Accessed 14 April 2018

[24] GeoHealthCheck. http://geohealthcheck.org/. Accessed 14 April 2018

[25] Swagger. https://swagger.io/. Accessed 14 April 2018

[26] MapProxy. https://mapproxy.org/. Accessed 14 April 2018

[27] pycsw. http://pycsw.org/. Accessed 14 April 2018

[28] Celery. http://www.celeryproject.org/. Accessed 14 April 2018

[29] Solr vs RDBMS script. https://goo.gl/ZqHxog. Accessed 14 April 2018

Abbreviations

API: Application programming interface; CGA: Center for geographic analysis; CMS: Content management system; CSW: Catalogue service for the web; DCMS: Data content management system; GeoCMS: Geospatial content management system; HHypermap: Harvard hypermap; IQSS: Institute for quantitative social science; JSON: JavaScript object notation; MIT: Massachusetts institute of technology; OGC: Open geospatial consortium; RDBMS: Relational database management system; REST: Representational state transfer; SDI: Spatial data infrastructure; TMS: Tile map service; WCS: Web coverage service; WFS: Web feature service; WFS-T: Web feature service - transaction; WMS: Web map service; WMS-C: Web map service tile caching; WMTS: Web map tile service

Acknowledgements

The authors thank all the contributors to the Hypermap, WorldMap and GeoNode platform source code, particularly: Matt Bertrand, Simone Dalmasso, Alessio Fabiani, Jorge Martínez Gómez, Wendy Guan, Jeffrey Johnson, Devika Kakkar, Jude Mwenda, Ariel Núñez, Luis Pallares, David Smiley, Charles Thao, Mingda Zhang

Funding

This work is partially funded by the U.S. National Endowment for the Humanities, Digital Humanities Implementation Grant #HK5009113 and the U.S. National Science Foundation's Industry-University Cooperative Research Centers Program (IUCRC) grant for the Spatiotemporal Thinking, Computing, and Applications Center (STC) #1338914. Grant administration supported by the Institute for Quantitative Social Science, Harvard University.

Authors' contributions

PC wrote the manuscript. ATK and BL reviewed the manuscript. PC and ATK are members of the Hypermap, GeoNode and pycsw development team. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Author details

[1]Center for Geographic Analysis, Harvard University, 1737 Cambridge Street, K350, 02138 Cambridge, MA, USA . [2]Open Source Geospatial Foundation, 14525 Sw Millikan Way, 24105 Beaverton, OR, USA .

Received: 20 December 2017 Accepted: 1 May 2018
Published online: 11 June 2018

References

1. Li W, Yang C, Yang C. An active crawler for discovering geospatial web services and their distribution pattern–a case study of ogc web map service. Int J Geographical Inf Sci. 2010;24(8):1127–47.
2. Xiujun M, Gang L, Kunqing X, Meng S. A peer-to-peer approach to geospatial web services discovery. In: Proceedings of the 1st international conference on Scalable information systems. New York: ACM; 2006. p. 53.
3. Sample JT, Ladner R, Shulman L, Ioup E, Petry F, Warner E, Shaw K, McCreedy FP. Enhancing the us navy's gidb portal with web services. IEEE Internet Comput. 2006;10(5):53–60.
4. Schutzberg A. Skylab mobilesystems crawls the web for web map services. OGC user. 2006;8:1–3.
5. McCandless M, Hatcher E, Gospodnetic O. Lucene in action: covers Apache Lucene 3.0. Birmingham: Manning Publications Co.; 2010.
6. Wu S, Zhang M, Huang Q, Wan C, Cao J, Gui Z, Qin K, et al. Design a web portal for visualizing and exploring service quality of global ogc web map services. In: Geoinformatics, 23rd International Conference on. Wuhan: IEEE; 2015. p. 1–5.
7. Shadbolt N, O'Hara K, Berners-Lee T, Gibbins N, Glaser H, Hall W, et al. Linked open government data: Lessons from data. gov. uk. IEEE Intell Syst. 2012;27(3):16–24.
8. King G. An introduction to the dataverse network as an infrastructure for data sharing. Sociol Methods Res. 2007;36(2):173–99.
9. de La Beaujardiere J. Opengis® web map server implementation specification: Open Geospatial Consortium Inc, OGC; 2006, pp. 06–042. http://portal.opengeospatial.org/files/?artifact_id=14416.
10. Baumann P. Ogc wcs 2.0 interface standard—core. Wayland: Open Geospatial Consortium; 2010.
11. Guan WW, Bol PK, Lewis BG, Bertrand M, Berman ML, Blossom JC. Worldmap–a geospatial framework for collaborative research. Ann IS. 2012;18(2):121–34.
12. Corti P, Lewis B. Making temporal search more central in spatial data infrastructures. In: ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences 4. Gottingen: Copernicus GmbH; 2017.
13. Corti P, Lewis BG, Kralidis T, Mwenda J. Implementing an open source spatio-temporal search platform for spatial data infrastructures. Tech. rep., PeerJ Preprints. 2016.