Generic XML-based Framework for Metadata Portals [⋆]

Uwe Schindler [∗], Michael Diepenbroek

Center for Marine Environmental Sciences (MARUM), University of Bremen, Leobener Straße, D-28359 Bremen, Germany

doi:10.1016/j.cageo.2008.02.023

Received 25 June 2007; revised 2 February 2008; accepted 9 February 2008

Abstract

We present a generic and flexible framework for building geoscientific metadata portals independent of content standards for metadata and protocols. Data can be harvested with commonly used protocols (e.g., Open Archives Initiative Protocol for Metadata Harvesting) and metadata standards like DIF or ISO 19115. The new Java-based portal software supports any XML encoding and makes metadata searchable through Apache Lucene. Software administrators are free to define searchable fields independent of their type using XPath. In addition, by extending the full-text search engine (FTS) Apache Lucene, we have significantly improved queries for numerical and date/time ranges by supplying a new trie-based algorithm, thus enabling high-performance space/time retrievals in FTS-based geo portals. The harvested metadata are stored in separate indexes, which makes it possible to combine these into different portals. The portal-specific Java API and web service interface is highly flexible and supports custom front-ends for users, provides automatic query completion (AJAX), and dynamic visualization with conventional mapping tools. The software has been made freely available through the open source concept.

Key words: spatial data infrastructure, metadata portal, metadata standard, open archives, full-text search, Apache Lucene

[⋆] Program code available open source at `http://www.panFMP.org/`

[∗] Corresponding author.
Email addresses: `uschindler@pangaea.de` (Uwe Schindler), `mdiepenbroek@pangaea.de` (Michael Diepenbroek).

1 Introduction

1.1 Background and Motivation

Complex and large-scale investigations carried out within many projects in the fields of earth and biological sciences have produced a huge amount of observational and modelling data with extensive geographical and temporal coverage. During the past decade there have been various initiatives and approaches in networking global data services and related data stored in distributed archives. The recently formed Group on Earth Observations (GEO) conceived a plan for a Global Earth Observations System of Systems (GEOSS, see Battrick, 2005), which largely builds on the principles of Global Spatial Data Infrastructures (GSDI, see Nebert, 2004). Correspondingly, the Infrastructure for Spatial Information in Europe (INSPIRE, see The European Parliament and Council, 2007) is an EU directive to foster the coordination of geodata and the interoperability of data services within Europe.

These two initatives clearly emphasize the need for portal frameworks to provide simple and transparent access to geoscientific data and metadata. In fact, global standardization efforts have been quite successful in recent years, leading to some convergence in developments. Nevertheless, portal developers have to cope with two problems: First, the fact that data providers have different backgrounds and capabilities and furnish data and metadata using various protocols and content structures. In practice, puristic, homogeneous approaches based on unique standards are likely to cause exclusion of at least part of the potential data sources and – due to the dynamics in SDI development – might result in a premature end of operation. Therefore, portal implementations are needed that compensate for the heterogeneity of the standards employed. Second, search engines like Google have changed the way scientist are searching for publications and data. Users are accustomed to a single input line to enter search terms, and expect fast response times in receiving results ranked by their relevance. This requires a change from relational databases, which are commonly used in geosciences for data retrieval, to full-text search (FTS) engines (Bennett, 2004). A major drawback of FTS engines, however, is the inability to perform numerical or date/time range retrievals, which are frequently needed when searching for limited temporal or spatial coverages (e.g., for geographic bounding boxes). At present, these types of queries are the strength of relational databases (which have limited full-text possibilities). Therefore, FTS implementations are needed that support fast and effective numerical range queries.

1.2 State of the Art

A metadata portal is an online internet site that typically provides simple and transparent access to distributed information resources. Metadata portals in the geoscientific world are also known as "Spatial Data Directory", "Clearinghouse", "Geospatial One-Stop Portal" (Nebert, 2004, p. 40), or "Geo Portal" (Maguire and Longley, 2005). In the scientific context, it is perceived as a broker among a multitude of data centers, information systems, institutions, organizations, and the scientific community (see Abad-Mota, 2001; Maguire and Longley, 2005). Technically speaking, it is designed to use distributed applications, different numbers and types of middleware and hardware, to facilitate the discovery and mining of data from a number of different sources.

1.2.1 Metadata

Metadata are data about data. They provide basic information about data and the information can help archiving, discovering, and describing data. In general, in the geosciences metadata at a minimum have to answer the questions: Who has measured, observed, or calculated what, where, when, and how? A number of metadata standards define the corresponding content structures for collecting metadata. The most important ones in the geographic information domain are ISO 19115 (Kresse and Fadaie, 2004), FGDC[1], DIF[2], and Dublin Core[3]. These *content standards* allow users to identify data sets not only by bibliographic information, such as authors, title, date, publisher etc.; they also make available spatial or temporal coverage (Dublin Core only as nominal values), parameters used, and data quality. For interoperability among data centers, these metadata records are often *encoded* into XML documents with well-defined schemas.

Having the compatibility of these major standards in at least the core fields, it is basically possible to map fields to a common content structure and thus to allow for consistent retrieval of metainformation. At present, however, existing portal software packages such as OJAX[4] only support single-content standards. An improved interoperability between standards would give added value to data providers and would foster the implementation of larger metadata networks and portals.

[1] Federal Geographic Data Committee. `http://www.fgdc.gov/metadata`
[2] Directory Interchange Format. `http://gcmd.nasa.gov/User/difguide/`
[3] The Dublin Core Metadata Initiative – DCMI Metadata Terms. `http://dublincore.org/documents/dcmi-terms/`
[4] Ajax powered metasearch service. `http://ojax.sourceforge.net/`

1.2.2 Search Technology for distributed Catalogues

Metadata portals allow users to search for data sets based on metadata schemas used by data providers. Current portals use two different approaches for metadata search: (1) searching on distributed catalogues or (2) harvesting catalogues into a central searchable catalogue.

In distributed search infrastructures, every data provider not only has his own metadata catalogue, but also a corresponding search interface to the portal (e.g., web service based). Search requests are sent to all data providers. The portal only needs to collect the search results from the providers, then rank and display these to the end user. Examples of this architecture are the NSDI Clearinghouse[5] (Nebert, 2000) and GeoPortal.BUND[6].

The implicit assumption behind this approach is that all data providers need to have interoperable interfaces for the search infrastructure (e.g., database software), thesauri, and catalogue service software. Due to the distributed architecture, the response of the end-user search interface is sluggish. The slowest search provider and the network connections dictate the overall response time. Another drawback is the need for special algorithms to merge and rank search results in a user-friendly way.

In the harvesting solution, every data provider has its own metadata catalogue but the search engine is centralized. The portal periodically harvests all metadata records into a central index and serves search requests from there. Major web search engines like Google or the FGDC related Geospatial One-Stop[7] (Goodchild et al., 2007) are based on this concept. The response time is optimal because only local components are used in the search process.

One disadvantage of this approach is that the search results are not necessarily current and up-to-date, because updates or deletes of the metadata records at the source are not immediately reflected in the portal. Synchronization is dependant on the harvesting frequency, and users may experience "404 Not Found" errors when accessing data that was deleted in the harvested repository. Furthermore, there is the need for additional data storage space on the portal's side. In the optimal case this data storage is directly embedded in the search index component.

Nevertheless, the disadvantages of a distributed search infrastructure outweigh those of the harvesting approach (Lossau, 2004) making harvesting a more efficient solution.

[5] `http://clearinghouse.fgdc.gov/`
[6] `http://www.geoportal.bund.de/`
[7] `http://www.geodata.gov/`

1.2.3 Protocols for Metadata Retrieval

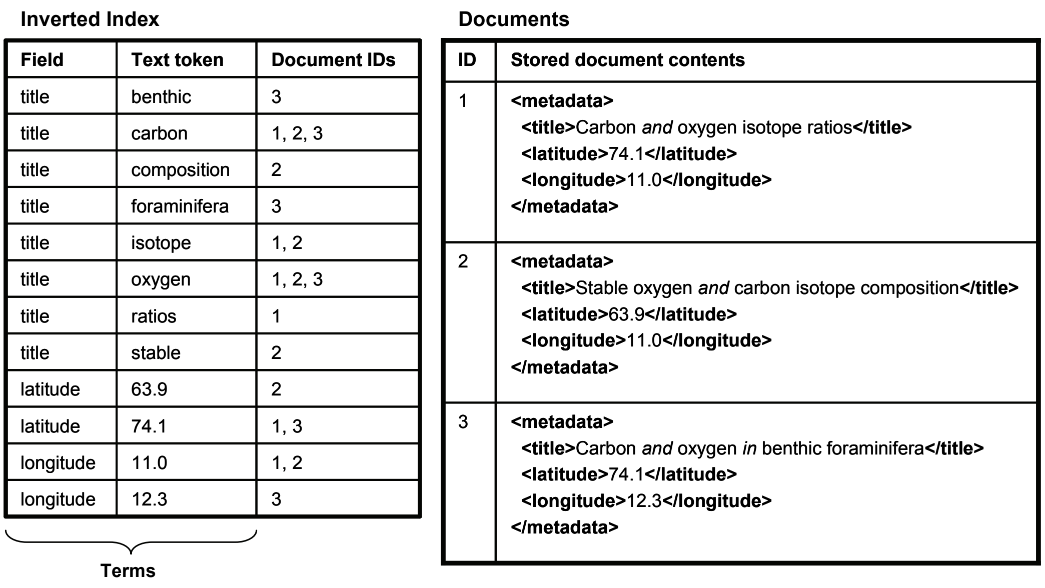
A standardized network protocol for distributed searches in the geoscientific world is OGC Catalogue Services (OGC CS-W / OGC CAT) [8] . Metadata portals can create search queries based on the metadata content standard used and receive search results in XML encoding. So far, however, CS-W has only limited support for the harvesting approach. CS-W supports harvesting of external documents for inclusion into the local catalogue [9] , but there is no special interface for external portals to harvest a CS-W. It may be possible to start a CS-W search request that returns metadata records to harvest, but it does not correctly support incremental updates (harvesting only new or changed documents), which is essential for large catalogues. Future developments of the OGC CS-W interface may make this possible.

A further network protocol – widely used in library environments – is Z39.50 [10] , which is a complex, binary pre-Web technology. Like CS-W, it does not natively support harvesting, but metadata according to Z39.50 GEO profile [11] can be extracted as XML files in FGDC content standard from the data stream.

In contrast, the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH, see Van de Sompel et al., 2004) [12] is a simple to use, HTTP-based protocol that even supports incremental harvesting. The protocol is widely used within the library world. During each harvesting event, data providers are asked for new or changed metadata records filtered out by the timestamp of the last harvesting operation. If the repository supports tracking of deleted datasets, the list of those is also provided during harvesting. If this feature is not supported by the repository, the catalogue must be completely re-harvested every time to sort out the outdated records.

Current implementations of OAI-PMH compliant service providers are available in the open source community for various programming languages (Java, PHP, PERL,...) and infrastructures (harvesting to file system, to database,...). Unfortunately, all of those implementations are limited to a single content standard (mostly Dublin Core metadata) and have a corresponding fixed data storage backend. Database structures in these software packages are designed for harvesting this simple metadata content standard and cannot be changed

[8] OGC Catalogue Services. `http://www.opengeospatial.org/standards/cat`
[9] an example of such a software is conterra's terraCatalog version 2.2. `http://www.conterra.de/en/products/sdi/terra/`
[10] ANSI/NISO. Information Retrieval (Z39.50): Application Service Definition and Protocol Specification. `http://www.loc.gov/z3950/agency/`
[11] Z39.50 Application Profile for Geospatial Metadata or "GEO". `http://www.blueangeltech.com/standards/GeoProfile/geo22.htm`
[12] `http://www.openarchives.org/OAI/openarchivesprotocol.html`


Inverted Index

Field	Text token	Document IDs
title	benthic	3
title	carbon	1, 2, 3
title	composition	2
title	foraminifera	3
title	isotope	1, 2
title	oxygen	1, 2, 3
title	ratios	1
title	stable	2
latitude	63.9	2
latitude	74.1	1, 3
longitude	11.0	1, 2
longitude	12.3	3

Terms

Documents

ID	Stored document contents
1	**<metadata>** **<title>**Carbon *and* oxygen isotope ratios**</title>** **<latitude>**74.1**</latitude>** **<longitude>**11.0**</longitude>** **</metadata>**
2	**<metadata>** **<title>**Stable oxygen *and* carbon isotope composition**</title>** **<latitude>**63.9**</latitude>** **<longitude>**11.0**</longitude>** **</metadata>**
3	**<metadata>** **<title>**Carbon *and* oxygen *in* benthic foraminifera**</title>** **<latitude>**74.1**</latitude>** **<longitude>**12.3**</longitude>** **</metadata>**

Fig. 1. Example of inverted index as implemented by Apache Lucene: Document fields (right side) are tokenized (without *stop-words*) and stored in inverted index (left side). If user searches for term (text token in one of the fields), document IDs are returned and search results may be displayed using stored contents.

without major changes in the software.

1.2.4 Search Engines

The harvesting approach requires central storage and searching of the collected metadata on the portal side. Bennett (2004) compares the search features of full-text search engines (FTS) with relational database management systems (RDBMS) and comes to the conclusion that FTS engines are better adapted to the search requirements for text-based information inventories. Even conventional combinations of MySQL[13] and built-in FTS, for example, are limited in functionality and performance, especially for larger databases. Moreover, because the editing system of the metadata records stays at the data providers, there is no need for a relational database. The central facet of a portal is searching and not curation of metadata.

A widely used open source FTS is Apache Lucene. It is a high-performance, full-featured text search engine library written in Java and suitable for nearly any application requiring full-text search, especially cross-platform (Hatcher and Gospodnetic, 2004). Its features include: variable query types like phrase queries or proximity queries, and the use of boolean operators. Furthermore, because Apache Lucene offers support for fielded searching, it is possible to search only on specific metadata parts. The segment-based index structure

[13] `http://www.mysql.com/`

enables searches on multiple indexes with merged results and also allows for simultaneous updates (compare with “transactions” in relational databases).

The fundamental concepts in Apache Lucene are “index”, “document”[14], “field”, and “term” (Bennett, 2004, table 1). An “index” contains a sequence of “documents”, which are a sequence of “fields”. Each field gets tokenized and “terms”, which are pairs of field name and text tokens, are generated. The index stores term origin and statistics in order to make term-based search more efficient. Apache Lucene’s index falls into the family of indexes known as an inverted index (Harman et al., 1992). This is because it can list – for a term – the documents that contain it. This is the inverse of the natural relationship, in which documents list terms. Additionally, untokenized field contents may be additionally stored for later display. Figure 1 shows an example of three indexed documents and the corresponding inverted index.

A general disadvantage of FTS – compared to RDBMS – is its limited performance with range queries for date/time or numerical values, which are essential elements for metadata portals, because users need to be able to set search constraints on temporal or spatial coverage.

2 panFMP – A Generic XML-based Metadata Portal Software Framework

Based on the needs of scientific communities we have designed a generic portal system architecture suitable for metadata portals in Earth and biological sciences without constraints on the metadata content standard used. This new Java-based portal software supports any XML encoding that can be harvested from OAI-PMH repositories, file systems, or web servers and makes them searchable through Apache Lucene without any other database software.

Figure 2 shows the main components of the package, which consists of (1) a harvester and index builder component, that collects metadata from the providers (see section 2.1), (2) Apache Lucene as central indexing component and data store, (3) a search interface for querying indexes (see section 2.3), and (4) configuration of metadata formats, data providers and searchable fields in an XML-based configuration file (see section 2.4). To support fast queries with temporal or spatial coverages (e.g., for geographic bounding boxes), a trie-based implementation of range queries was created (see section 2.2).

[14] elsewhere in this paper, we use the term “metadata record” instead of “document” because it more accurately describes the scope. “Document” comes from the original purpose of FTS engines: searching in text documents.

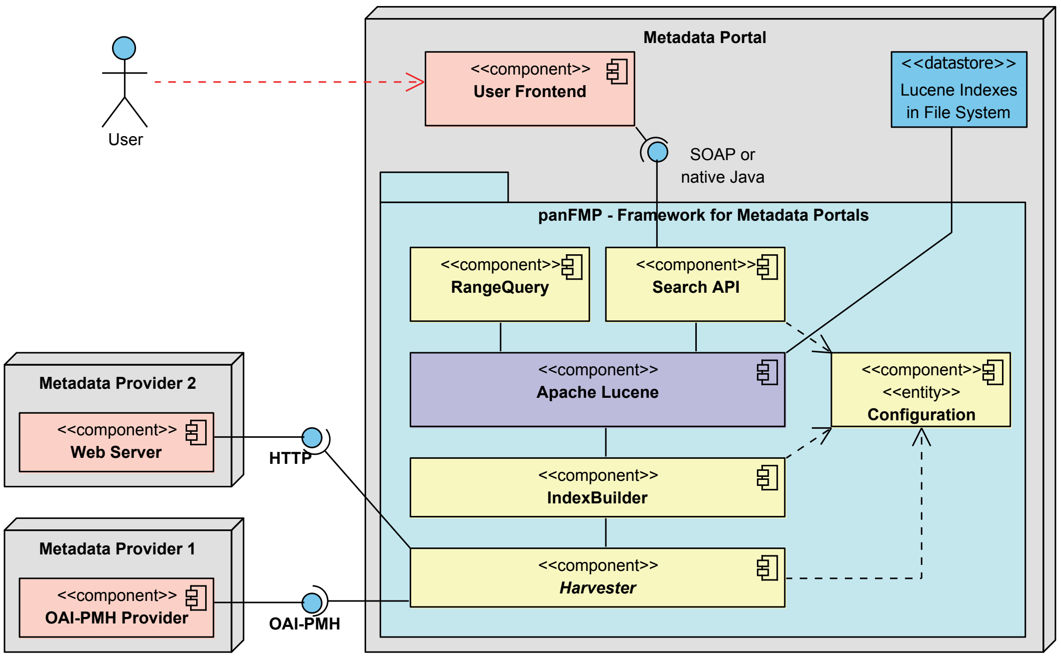

Metadata Portal
User
<<component>> User Frontend
<<datastore>> Lucene Indexes in File System
SOAP or native Java
panFMP - Framework for Metadata Portals
<<component>> RangeQuery
<<component>> Search API
<<component>> Apache Lucene
<<component>> <<entity>> Configuration
<<component>> IndexBuilder
<<component>> Harvester
Metadata Provider 2
<<component>> Web Server
HTTP
Metadata Provider 1
<<component>> OAI-PMH Provider
OAI-PMH

Fig. 2. Overview of all components needed for metadata portal: The portal framework "panFMP" described in this paper is cyan colored and consists of Apache Lucene (purple) and components developed by the authors for harvesting, indexing, configuration, search (yellow). Parts that must be supplied by portal developers or third parties are colored red.

Portal developers need to generate a configuration file defining index properties, harvestable metadata providers, and searchable fields. After initial harvesting into the Apache Lucene indexes located in the local file system, it is possible to query those using the application programming interface (API) supplied. For that a web-based user interface may be created.

The portal framework has been made freely available through the open source concept under the Apache License[15] and is hosted by SourceForge.net[16]. Version 1.0 will be officially released at `http://www.panFMP.org/` when the code base has proven its usability and the design of the programming API for portal implementers is stable.

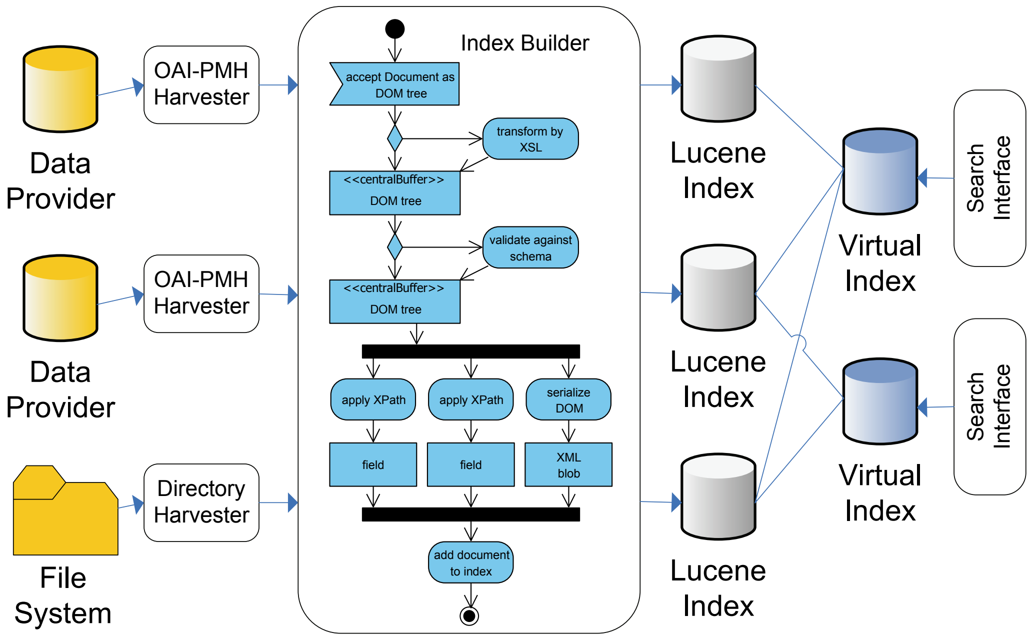


Fig. 3. Overview of harvester and index builder

2.1 Harvester and Index Builder

The portal software harvests all metadata into the Apache Lucene index directly without the need to store them separately (see fig. 3). The event-based abstract harvester class is universally designed to support a number of different harvesting solutions. It is responsible for collecting new or updated metadata XML files from various sources. For each new or updated metadata record it creates a DOM[17] tree and notifies the index builder, which then analyzes the tree and updates the index in a different thread(s).

Additionally, the harvester class allows for an on-the-fly transformation by XSLT[18] from any metadata content standard into the index specific one – provided that the content standards are compatible – and allows for validation of resulting metadata files by an XML schema. These two features allow the harvest of metadata into unique index structures from data providers that supply metadata in various content standards. This is essential for a comprehensive portal with many different data providers.

[15] Apache License, Version 2.0, January 2004. `http://www.apache.org/licenses/LICENSE-2.0`
[16] SourceForge.net is the world's largest Open Source software development web site, hosting more than 100 000 projects and over a million registered users with a centralized resource for managing projects, issues, communications, and code. `http://sourceforge.net/`
[17] Document Object Model. `http://www.w3.org/DOM/`
[18] XSL Transformations. `http://www.w3.org/TR/xslt`

The reference implementation is a high-performance OAI-PMH harvester (class `OAIHarvester`). We opted for our own implementation because available open-source harvesters support only one distinct metadata scheme, usually Dublin Core, because the data-storage backend (e.g., relational databases) is fixed and Dublin Core is a minimal requirement for OAI-PMH. In contrast, metadata from science is often complex (e.g., ISO 19115). Because OAI-PMH embeds all records in a large XML file that is parsed sequentially, available DOM-based harvesters often fail with "out of memory" problems. We use SAX[19] (which works sequentially and is event-based) to parse the OAI response using Digester[20] as the frontend. Digester was extended to support on-the-fly switching to build a DOM tree when coming to the metadata component and switching back when going further with OAI-PMH protocol. This makes it possible to sequentially harvest the entire XML file and build multiple separate DOM trees that are sent to the index builder. To also support less complex setups, our package contains a very simple harvester for XML files from local file systems (`DirectoryHarvester`). Another harvester (`WebCrawlingHarvester`) can be employed to an HTML web page (e.g., a directory listing) containing links to XML files for harvesting.

Due to the abstract harvester design, it is possible to extend scope of its capability, making it possible to include implementations for other protocols such as future versions of CS-W or even Z39.50[21].

All harvested documents are fed to the index builder. During this step it is possible to filter unwanted documents by checking for characteristics using an XPath[22] function that returns a boolean value. The index builder then extracts the contents from the DOM tree and passes four types of objects to the full-text search engine, Apache Lucene, which performs the subsequent indexing. The four types of objects are:

- **A list of user-defined fields with their contents.** The open architecture makes it possible to define all searchable fields in several data formats using XPath and XSL templates syntax. Not only does this allow full-text queries, it also means that numerical or date ranges are retrievable on specific parts of the metadata. In Apache Lucene, fields may be stored, in which case their text is stored in the index literally, in a non-inverted manner for later retrieval as part of the record. Fields that are inverted are called indexed. A field may be both stored and indexed. The text of a field may be tokenized into terms to be indexed, or the text of a field may be used literally as a

[19] Simple API for XML (Brownell, 2002). `http://www.saxproject.org/`
[20] Jakarta Commons Digester. `http://jakarta.apache.org/commons/digester/`
[21] Currently, metadata according to Z39.50 GEO profile can be extracted as FGDC XML files by third party tools to make them ready for harvesting by `DirectoryHarvester`.
[22] XML Path Language. `http://www.w3.org/TR/xpath`

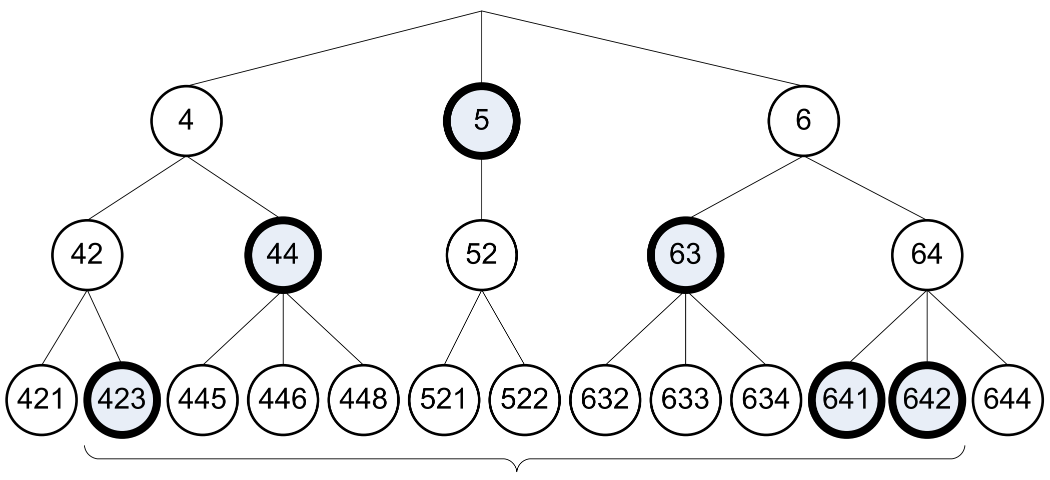


Fig. 4. Example on trie-based recursive splitting of range query with three precisions (simplified for demonstration): User wants to find all records with terms between "423" and "642". Instead of selecting all terms in lowermost row, query is optimized to only match on labelled terms with lower precision, where applicable. It is enough to select term "5" to match all records starting with "5" ("521", "522") or "44" for "445", "446", "448". Query is therefore simplified to match all records containing terms "423", "44", "5", "63", "641", or "642".

term to be indexed. Most fields are tokenized, but it is useful for identifier and numeric fields to be indexed literally. The implementation of searchable (inverted) numerical fields (even dates are numerical values) inside a full-text index is described in section 2.2. XPath/XSL template definitions and properties of fields are achieved by the configuration file, which is described in section 2.4.

- **A tokenized default field covering the entire record for a Google-like search.**
- **The full string-serialized DOM tree as a compressed stored field.** It comprises the central metadata inventory to be used, for example, for the display of metadata details.
- **Control information for each record:** a timestamp and the record identifier for later updates.

The metadata of all the various harvested data providers are stored internally as separate indexes (with exactly equal structure) giving the administrator the possibility to manage them separately and allowing for flexible combinations into "virtual" indexes for searching.

2.2 Optimized Range Queries

Because Apache Lucene is a full-text search engine and not a conventional database, it cannot handle numerical ranges (e.g., field value is inside user defined bounds, even dates are numerical values). So it expands a range to a large "OR" query consisting of all terms between the boundaries (this is called query rewriting and is also used for wildcard queries). When the index contains a lot of records with distinct numerical values and the range boundaries are far-off, this "OR" list is extremely long. Older versions of Apache Lucene ($<$ 2.1, current is version 2.3) were limited to a maximum number of "OR" terms and threw exceptions. Current versions use a bitmap for these type of queries, nevertheless, all terms between the range boundaries must be discovered.

We have developed an extension to Apache Lucene that stores the numerical values in a special string-encoded format with variable precision (all numerical values like doubles, longs, and timestamps are converted to lexicographic sortable string representations of `long long words` and stored with precisions from one byte to the full 8 bytes). This is similar to a "trie memory" [23], as was originally proposed by Fredkin (1960). A range is then divided recursively into multiple intervals for searching (see fig. 4): The center of the range is searched only with the lowest possible precision in the trie, the boundaries are matched more exactly (using the idea from the paper of de la Briandais, 1959). This reduces the number of terms dramatically (in the lowest precision of 1-byte the index only contains a maximum of 256 distinct values). Overall, a range could consist of a theoretical maximum of

$$\underbrace{7 \times 255}_{\substack{\text{boundaries split into}\\ \text{7 different precisions}}} \times \underbrace{2}_{\substack{\text{lower and}\\ \text{upper part}}} + \underbrace{255}_{\substack{\text{center with}\\ \text{lowest precision}}} = 3\,825$$

distinct terms (when there is a term for every distinct value of an 8-byte-number in the index and the range covers all of them; a maximum of 255 distinct values is used because it would always be possible to reduce the full 256 values to one term with degraded precision). In practise, we have seen up to 300 terms in most cases (index with 500 000 metadata records and a homogeneous dispersion of values).

This dramatically improves the performance of Apache Lucene with range queries, which is no longer dependent on the index size and number of distinct values because there is an upper limit not related to any of these properties.

2.3 Search Interface

We added a portal-specific Java API to the existing one of Apache Lucene, which allows for a full-featured and flexible usage of the search engine. It

[23] "Trie" is derived from "reTRIEval", also known as "prefix tree".

abstracts the underlying Lucene API and combines it with the portal configuration file (cf. section 2.4) using a “factory” to create correctly configured Apache Lucene `Query` objects. In this way, portal developers can formulate queries as described in section 1.2.4 using the defined fields. It uses optimized range queries (see section 2.2) for queries on numerical and date/time values. Search results are returned in the XML metadata encoding and/or the stored fields. An API providing query auto completion was implemented, thus giving users guidance to retrievable terms. This improves the overall ergonomy of the web frontend. Portal developers can implement this by using AJAX[24] technologies.

Based on the API the programmer can even display the results in maps (e.g., using UMN MapServer[25] or Google Earth[26]). Sample implementations for various metadata content standards are provided together with the portal package.

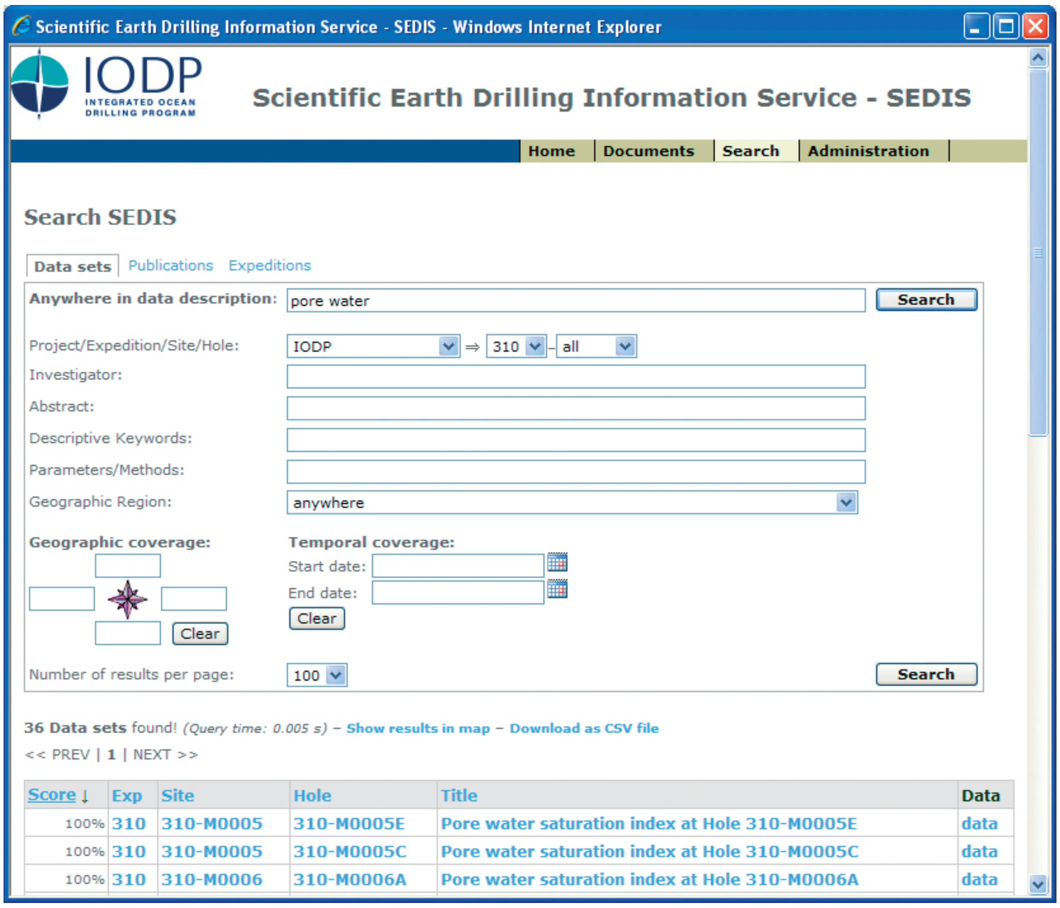
For portals running in environments without Java support in the web server, a simplified version of the API can be made available as a web service to SOAP/WSDL clients.

2.4 Configuration of Metadata and Indexes

Many components of the portal software are highly customizable. We developed an XML-based configuration file that handles definition of the metadata format, data providers including their properties and, last but not least, the search interface. The file is read by the harvester and search interface on startup and parsed with Jakarta Commons Digester. We selected XML as the preferred file encoding because it harmonizes perfectly with the metadata to be harvested. As described in section 2.1, index fields are defined by XPath queries on the underlying metadata file. An XML-based configuration file makes it possible to handle XML namespace[27] declarations used by the XPath queries in a very elegant way (like it is done by XSLT). Optional metadata transformations can also be expressed by XSLT embedded in the configuration file.

The definition of harvester properties (attributes and type of metadata repository), the stream analyzer used (for tokenization of string fields during har-

[24] Asynchronous JavaScript and XML (AJAX). Several tools are available for enabling web sites with auto-completing input fields, e.g., “Yahoo! User Interface Library”, `http://developer.yahoo.com/yui/`
[25] `http://mapserver.gis.umn.edu/`
[26] `http://earth.google.com/`
[27] Namespaces in XML. `http://www.w3.org/TR/xml-names`


Scientific Earth Drilling Information Service - SEDIS - Windows Internet Explorer
IODP
INTEGRATED OCEAN DRILLING PROGRAM
Home
Documents
Search
Administration
Data sets
Publications
Expeditions
Search
Clear
Clear
Search

Scientific Earth Drilling Information Service - SEDIS

Search SEDIS

Anywhere in data description:	pore water
Project/Expedition/Site/Hole:	IODP ⇒ 310 - all
Investigator:	
Abstract:	
Descriptive Keywords:	
Parameters/Methods:	
Geographic Region:	anywhere

Geographic coverage:

Temporal coverage:
Start date:
End date:

Number of results per page: 100

36 Data sets found! *(Query time: 0.005 s)* – **Show results in map** – **Download as CSV file**

<< PREV | **1** | NEXT >>

Score ↓	Exp	Site	Hole	Title	Data
100%	310	310-M0005	310-M0005E	Pore water saturation index at Hole 310-M0005E	data
100%	310	310-M0005	310-M0005C	Pore water saturation index at Hole 310-M0005C	data
100%	310	310-M0006	310-M0006A	Pore water saturation index at Hole 310-M0006A	data

Fig. 5. SEDIS data portal web interface

vesting and parsing search queries), and index configurations (which index contains what repository, which indexes can be searched as virtual index,...) are also supplied by the configuration.

3 Usage Scenarios

Based on the portal framework described in this article the World Data Center for Marine and Environmental Sciences (WDC-MARE) with its information system PANGAEA® (Diepenbroek et al., 2002) has implemented several metadata portals for EU and international projects (e.g., IODP, EUR-OCEANS, CARBOOCEAN, etc.)[28].

An example of such a data portal is IODP SEDIS[29] (Scientific Earth Drilling Information Service for the Integrated Ocean Drilling Program, see Miville

[28] a current list of portals with corresponding web addresses can be found at `http://www.panFMP.org/front_content.php?idcat=346`
[29] Scientific Earth Drilling Information Service. `http://sedis.iodp.org/`

et al., 2006), which disseminates and publishes data and metadata about scientific ocean drilling, regardless of the origin or location of data. Metadata providers to SEDIS are the IODP implementing organizations (IOs) from the United States (USIO), Japan (CDEX), Europe with Canada (ESO), Lamont-Doherty Earth Observatory (LDEO) for bore hole data, and NOAA/NGDC for legacy data. Each provider uses its own data management system. SEDIS consists of a central metadata index based on the ISO 19115 standard and ISO 19139 for its XML implementation using "panFMP" as the portal framework. The IOs are harvested using OAI-PMH. SEDIS can be expanded at a later stage to include other scientific drilling data from continental drilling (ICDP) or further data providers related to IODP. Current work comprises the inclusion of a search engine on IODP related publications (also selected from different providers) and advanced data search, visualization, and mapping tools.

In principle, the components for SEDIS and its portal are the same as in figure 2. The front-end software was written in PHP[30] using the web service interface of "panFMP" (see fig. 5).

Furthermore, SEDIS contains a catalogue of all expeditions stored in a MySQL database. This catalogue is also included in the portal by exporting all expedition metadata from the relational database to XML files into the local file system and harvesting them using the `DirectoryHarvester` of the portal framework. Because the metadata files are not "real" ISO 19139 files (expedition metadata are more simple), the harvester uses the XSL transformation feature of the framework.

Various other groups have already expressed an interest in using the framework software (e.g., ICSU WDC System[31], IODE[32]). Recently, the main search engine of PANGAEA® (formerly "PangaVista") and the C3Grid[33] were also adapted to the generic portal software. It verifies the generic design because in the case of PANGAEA® it was implemented by plugging in a custom harvester that accesses the XML metadata directly from the editorial system in an SYBASE database (a graphical overview can be seen in fig. 3 of Schindler et al., 2005). In C3Grid the framework was embedded in a grid architecture largely based on GridSphere[34] and Globus Toolkit[35]. Here the software is used for discovery of data sets and available workflows. In principle, any ex-

[30] PHP: Hypertext Preprocessor. `http://www.php.net/`
[31] World Data Center System. `http://www.ngdc.noaa.gov/wdc/`
[32] The IOC's International Oceanographic Data and Information Exchange. `http://www.iode.org/`
[33] Collaborative Climate Community Data and Processing Grid. `http://www.c3grid.de`
[34] The GridSphere Portal Framework. `http://www.gridsphere.org/`
[35] The Globus Alliance. `http://www.globus.org/`

isting XML metadata management system may be extended by "panFMP" as search engine add-on. For e.g. the Geospatial One-Stop it could supply a significant improvement of search speed. Moreover, if the metadata inventory, which by January 2006 contained 10400 records (Goodchild et al., 2007), grows significantly, a separate, well-scaling FTS infrastructure might be necessary – at least from the user perspective.

The search speed for any query type is excellent because top-ranked results normally show up immediately without any noticeable delay, because FTS engines do not need to wait for the full query to complete like with RDBMS. This conforms to current user experience with major internet search engines. We ran some test queries on different machines with $\approx 500\,000$ metadata records. On an Opteron machine with four processor cores and 8 GByte RAM running 64 bit Linux, results mostly display in $\ll 0.05$ s. Comparisons with other machines, for example, a low-cost developer's machine with the same Lucene indexes, did not show any difference in response times. Because the portal framework does not modify the search capabilities of the underlying Apache Lucene engine, apart from optimized range queries, results from other benchmarks as shown on the Apache Lucene homepage[36] also apply to "panFMP". In general, the portal machine should be a multi-processor for handling simultanous harvesting, metadata parsing, index updates, and search requests from users (≥ 4 processor cores). Disk space scales linear to the number of documents. As described in section 2.2, the optimized range query algorithm is no longer dependent on the index size and number of distinct numerical values. Usage of system memory is better than linear to the number of metadata records, and linear to the number of parallel search requests. Portal developers should assign a maximum memory suitable for their needs to the Java virtual machine. As this memory is often > 2 GBytes, a 64 bit platform is preferred, particularly because it is possible to tune index access times by using memory-mapping techniques on a 64 bit platform.

4 Conclusions

The new generic portal software helps to provide fast and low-barrier access to scientific data based on XML metadata formats. Service providers are free to use various metadata content standards that can be transformed into uniquely structured indexes. The use of a full-text search engine – in contrast to classical relational data bases – conforms to the current user experience. For the first time the speed deficit in range query execution has been solved by using trie structures, thus enabling high-performance space/time retrievals in FTS-based

[36] Apache Lucene – Resources – Performance Benchmarks. `http://lucene.apache.org/java/docs/benchmarks.html`

geo portals.

The portal framework is applicable to a wide range of architectures, data providers, and content models, including the digital library world. The generic design of the framework allows for extension of its use to various other protocols and content standards, and assures wide interoperability among data bases and data grids in the context of the Open Access Initiative (Chan et al., 2002) and beyond.

Acknowledgment

The authors are indebted to Walter Hale (MARUM) who assisted through constructive comments and proofreading the English text.

References

Abad-Mota, S., 2001. Databases and portals for knowledge management. In: Digital Libraries and Virtual Workplaces: Important Initiatives for Latin America in the Information Age. Inter-American Agency for Cooperation and Development, Washington, DC, USA, pp. 201–210.

Battrick, B., 2005. Global Earth Observation System of Systems (GEOSS) 10-year implementation plan reference document. ESA (European Space Agency) Publications Division, 11 pp., `http://www.earthobservations.org/documents/10-Year%20Implementation%20Plan.pdf`, [accessed 21 April 2008].

Bennett, M., 2004. Contrasting relational and full-text engines. NIE (New Idea Engineering) Enterprise Search Newsletter 2 (9), article 1, `http://ideaeng.com/pub/entsrch/issue09/article01.html`, [accessed 21 April 2008].

Brownell, D., 2002. SAX2. O'Reilly, Sebastopol, CA, USA, 240 pp.

Chan, L., Cuplinskas, D., Eisen, M., Friend, F., 2002. Budapest Open Access Initiative. `http://www.soros.org/openaccess/read.shtml`, [accessed 21 April 2008].

de la Briandais, R., 1959. File searching using variable length keys. In: Proceedings of the Western Joint Computer Conference, New York. Vol. 15. pp. 295–298.

Diepenbroek, M., Grobe, H., Reinke, M., Schindler, U., Schlitzer, R., Sieger, R., Wefer, G., 2002. PANGAEA–an information system for environmental sciences. Computers & Geosciences 28 (10), 1201–1210, doi:10.1016/S0098-3004(02)00039-0.

Fredkin, E., 1960. Trie memory. Communications of the ACM 3 (9), 490–499.

Goodchild, M. F., Fu, P., Rich, P., 2007. Sharing geographic information: an assessment of the Geospatial One-Stop. Annals of the Association of American Geographers 97 (2), 250–266, doi:10.1111/j.1467-8306.2007.00534.x.

Harman, D., Baeza-Yates, R., Fox, E., Lee, W., 1992. Inverted files. In: Frakes, W. B., Baeza-Yates, R. (Eds.), Information Retrieval: Data Structures and Algorithms. Prentice-Hall, New Jersey, USA, pp. 28–43.

Hatcher, E., Gospodnetic, O., 2004. Lucene in Action. Manning Publications, Greenwich, CT, USA, 456 pp.

Kresse, W., Fadaie, K., 2004. ISO Standards for Geographic Information. Springer, Heidelberg, Germany, 322 pp.

Lossau, N., 2004. Search engine technology and digital libraries: libraries need to discover the academic internet. D-Lib Magazine 10 (6), doi:10.1045/june2004-lossau.

Maguire, D. J., Longley, P. A., 2005. The emergence of geoportals and their role in spatial data infrastructures. Computers, Environment and Urban Systems 29 (1), 3–14, doi:10.1016/j.compenvurbsys.2004.05.012.

Miville, B., Soeding, E., Larsen, H. C., 2006. Scientific Earth Drilling Information Service for the Integrated Ocean Drilling Program. Geophysical Research Abstracts 8, 05486.

Nebert, D. D., 2000. Building a geospatial data clearinghouse for data discovery and access. Statistical Journal of the United Nations Economic Commission for Europe 17 (2), 149–156.

Nebert, D. D. (Ed.), 2004. The SDI Cookbook, Version 2.0. Global Spatial Data Infrastructure Association, Technical Working Group Chair, 171 pp., `http://www.gsdi.org/gsdicookbookindex.asp`, [accessed 21 April 2008].

Schindler, U., Brase, J., Diepenbroek, M., 2005. Webservices infrastructure for the registration of scientific primary data. In: Rauber, A., Christodoulakis, S., Tjoa, A. M. (Eds.), Research and Advanced Technology for Digital Libraries. Vol. 3652 of Lecture Notes in Computer Science. Springer, pp. 128–138, doi:10.1007/11551362_12.

The European Parliament and Council, 2007. Directive 2007/2/EC of establishing an infrastructure for spatial information in the European Community (INSPIRE). Official Journal of the European Union L108, 1–14, `http://www.ec-gis.org/inspire/directive/l_10820070425en00010014.pdf`, [accessed 21 April 2008].

Van de Sompel, H., Nelson, M., Lagoze, C., Warner, S., 2004. Resource harvesting within the OAI-PMH framework. D-Lib Magazine 10 (12), doi:10.1045/december2004-vandesompel.