International Journal of Geo-Information

MDPI

Article

Near-Real-Time OGC Catalogue Service for Geoscience Big Data

Jia Song [1,2,3] and Liping Di [1,*]

[1] Center for Spatial Information Science and Systems, George Mason University, 4400 University Drive, MSN 6E1, Fairfax, VA 22030, USA; songj@lreis.ac.cn

[2] State Key Laboratory of Resources and Environmental Information System, Institute of Geographic Sciences and Natural Resources Research, Chinese Academy of Sciences, 11A, Datun Road, Chaoyang District, Beijing 100101, China

[3] Jiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, Nanjing 210023, China

* Correspondence: ldi@gmu.edu; Tel.: +1-703-993-6114

Received: 3 July 2017; Accepted: 26 October 2017; Published: 2 November 2017

Abstract: Geoscience data are typically big data, and they are distributed in various agencies and individuals worldwide. Efficient data sharing and interoperability are important for managing and applying geoscience data. The OGC (Open Geospatial Consortium) Catalogue Service for the Web (CSW) is an open interoperability standard for supporting the discovery of geospatial data. In the past, regular OGC catalogue services have been studied, but few studies have discussed a near-real-time OGC catalogue service for geoscience big data. A near-real-time OGC catalogue service requires frequent updates of a metadata repository in a short time. When dealing with massive amounts of geoscience data, this comprises an extremely challenging issue. Discovering these data via an OGC catalogue service in near real-time is desirable. In this study, we focus on how the near-real-time OGC catalogue service is realized through several lightweight data structures, algorithms, and tools. We propose a framework of a near-real-time OGC catalogue service and discuss each element of the framework to which more attention should be paid when dealing with the massive amounts of real-time data, followed by a review of several methods that need to be considered in a near-real-time OGC CSW service. A case study on providing an OGC catalogue service to Unidata real-time data is presented to demonstrate how specific methods are utilized to deal with real-time data. The goal of this paper is to fill the gap in knowledge regarding an OGC catalogue service for geoscience big data, and it has realistic significance in facilitating a near-real-time OGC catalogue service.

Keywords: CSW; catalogue service; big data; Unidata; metadata; real time

1. Introduction

Geoscience requires significant amounts of data and models for analyzing the past, current, and future geological status of the Earth. It also provides huge volumes of data in recording Earth observations from the past into the future. In particular, with the development of Earth observation (EO) technology, massive amounts of Earth observation data are continuously generated daily. There is no doubt that geoscience data are big data, and they are distributed in various agencies and individuals worldwide. Efficient data sharing and interoperability are important for managing and applying geoscience data.

The Open Geospatial Consortium (OGC), which is one of the international organizations dedicated to geospatial interoperability, has released various open standard implementations. The OGC Catalogue Service for the Web (CSW) [1] is one such important standard. It defines a standardized interface and metadata information model for the discovery of geospatial data. Metadata is data about

data. They are the entities of a catalogue. The OGC CSW interface includes discovery operations and transactional (e.g., insert/update/delete) operations. It is applied in various scenarios, including software tools, data portal services, and model workflow services. It has also been selected by many multi-institution organizations, such as GEO (Group on Earth Observation) and CEOS (Committee on Earth Observation Satellite) as a catalogue interface standard for sharing satellite-recorded Earth observation data.

In the past several years, OGC CSW applications have been developed to provide discovery services for several fundamental, global Earth observation data sources [2–7]. The NASA (National Aeronautics and Space Administration) Earth Observing System (EOS) Clearinghouse (ECHO), which enables the science community to discover and access NASA's data and services at the granule level, and provides a spatial and temporal metadata registry and order broker [8]. As ECHO uses their own metadata model and catalogue, a specific wrapper was developed for NASA ECHO to provide an OGC CSW service [3]. NOAA (National Oceanic and Atmospheric Administration) GOES (Geostationary Operational Environmental Satellite) and POES (Polar-orbiting Operational Environmental Satellite system) data, which are archived in NOAA's CLASS (Comprehensive Large Array-data Stewardship System), are also provided with an OGC CSW service developed by the Center for Spatial Information Science and System (CSISS) at George Mason University (Virginia, USA) under the support of an NOAA grant [6]. The Global Earth Observation System of Systems (GEOSS) [9], which is one of GEO's missions, has been proposed to facilitate global sharing and utilization of Earth observation (EO) data. An OGC catalogue service for the GEOSS AIP-2 (Architecture Implementation Pilot phase 2) polar ecosystem scenario was implemented by CSISS [2]. In addition, the CEOS WGISS (Working Group on Information Systems and Services) Integrated Catalogue (CWIC), which is a federated catalogue service, was proposed and implemented to discover geospatial data from multiple data centers [7]. In Europe, datasets and data services from the British Geological Survey (BGS) are discoverable through an OGC CSW service, which provides access to BGS ISO19115:2003 metadata.

These OGC CSW studies are the practices on the volume and variety characteristics in terms of the big data concept. Big data are commonly characterized by volume, variety, velocity, and veracity [10–12]. However, the velocity characteristic, which implies that the big data are often available in real time or near real time, has been ignored when providing an OGC CSW service in past studies or cases. These OGC CSW applications assume that the metadata repository, which is the essential part of a catalogue service, is not updated very frequently. In addition, it can be imagined that metadata registration for massive geoscience data is very time consuming. A near-real-time OGC catalogue service requires frequent updates to the metadata repository in a short time. When dealing with massive amounts of geoscience data from heterogeneous data sources, this poses a significant challenge. In reality, there are many forms of data with velocity characteristics, such as weather radar data, which is commonly generated every several minutes, and is factored into the weather forecast within a few hours. Thus, discovering these data with an OGC catalogue service in real time or near real time is desirable.

The object of this paper is to fill the knowledge gap regarding how a near real-time OGC catalogue service can be realized through several lightweight data structures, algorithms, and tools. A case study on providing an OGC catalogue service to Unidata real-time data is given in this paper as well. Unidata is a diverse community for sharing geoscience data and software tools, and supports Earth-system education and research [13]. It is a primary source of real-time atmospheric science data, and is supported by the United States and several countries outside the United States [14]. Providing a near-real-time OGC CSW service for the Unidata data will greatly facilitate atmospheric data sharing and interoperability with other geospatial data or models as well. The rest of this paper is organized as follows. In Section 2, we provide a framework for the near-real-time OGC catalogue service and discuss each element of the framework to which more attention should be paid when dealing with the massive amounts of near-real-time data, followed by some methods that need to be considered in a near-real-time OGC CSW service. In Section 4, we give an application case to demonstrate how

specific methods are utilized to deal with Unidata's real-time atmosphere data. Finally, a discussion and conclusions appear in Section 5.

2. Framework of the Near-Real-Time OGC Catalogue Service

The OGC catalogue service (CSW) specification defines a standardized interface and metadata information model for the discovery of geospatial data. A CSW server and metadata repository are the elements of a regular OGC catalogue service. Obviously, they are not enough to serve the needs of geoscience big data; furthermore, elements for updating metadata in near real time are additional essential elements, and metadata indigestion elements are indispensable as well. We propose the framework of the near-real-time OGC catalogue service as shown in Figure 1, and elements of the framework are described in the following subsections.

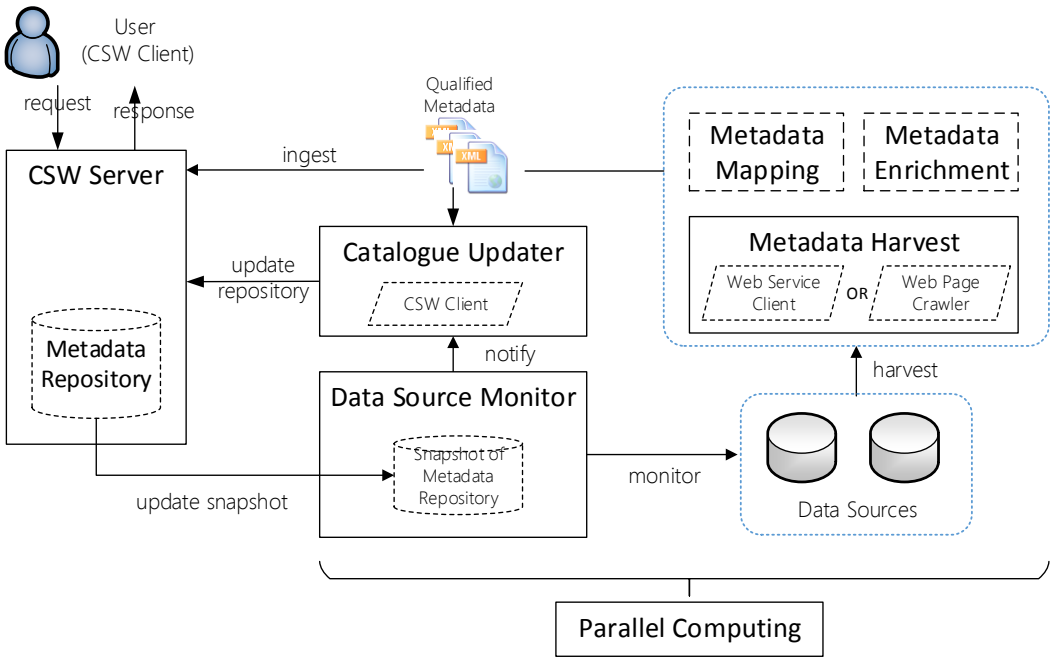


Figure 1. Framework of the near-real-time OGC catalogue service.

2.1. Fundamental Elements

As mentioned above, a CSW server and metadata repository are two fundamental elements of a regular OGC catalogue service. The CSW server provides a set of XML (Extensible Markup Language)-based service interfaces based on the HTTP server for supporting discovery, access, and registration operations for geospatial information resources; the metadata repository commonly employs a database for storing geospatial metadata. In addition, OGC also released available metadata models. They are called profiles, including the ebRIM [15,16] profile and the ISO 19115 profile [17], and meet the specific needs of different communities. These profiles (i.e., metadata models) can also be extended to cover other information.

2.2. Elements for Updating Metadata in Near Real Time

In order to discover geospatial data or model resources using an OGC CSW service, metadata must be registered and pushed into the metadata repository using an OGC CSW service. For real-time or near-real-time geoscience data, the metadata repository requires frequent updates to maintain accordance with data sources. Considering the velocity and volume characteristics of geoscience big data, the amount of metadata to be updated could be extremely massive and the update frequency is very high. In this situation, parallel processing approaches have to be considered. In addition,

the catalogue updater and data source monitor, which are indispensable elements for a near-real-time catalogue service, should be executed in parallel.

The catalogue updater is responsible for registering new metadata into the metadata repository. Since some data sources only archive the latest data and will remove relatively old data periodically, the catalogue updater also needs to be able to remove the metadata from the metadata repository. We have therefore designed another element, namely a "data source monitor", to find newly added or removed data archived in data sources. Unlike an entire catalogue update, which removes all metadata each time and then re-registers the entire corpus of current metadata, the incremental catalogue update only updates newly added or removed data. It can be seen that the entire catalogue is more easily implemented since it does not require picking out the newly added or removed data, but it is not suitable for the frequent big data updates because of the time cost. Therefore, we must choose a method of incremental catalogue updating targeted at the big data, and the data source monitor is indispensable for such real-time or near-real-time metadata updates.

2.3. Metadata Ingestion Elements

Metadata ingestion involves metadata harvesting, mapping, or enrichment if necessary. The metadata harvest element relies on the data source. If a data source provides the web service for harvesting their metadata, a metadata harvest client must be developed in terms of the web service provided by the data source. If a data source does not provide the web service and only publishes their data through web pages, a metadata web page crawler is required to grab and parse the web pages to get the metadata. The metadata mapping element is required when the harvested records do not follow the metadata profile of an OGC catalogue service provider. The metadata enrichment element is used to clean or add value to the metadata (e.g., normalization of punctuation, geocoding, etc.). The metadata mapping element and enrichment element are not mandatory when providing an OGC catalogue service. They are required only in those occasions in which the harvested metadata records are not compatible with the metadata model of the catalogue providers.

In summary, compared with a regular OGC catalogue service, the near-real-time OGC catalogue service not only involves building a CSW server and metadata repository, but also involves designing and implementing an efficient catalogue updater and data source monitor for updating metadata in near real time. In addition, regarding the metadata harvest element, whether a web service client or web page crawler, efficiency must be considered as well, and a catalogue update in parallel would be extremely helpful in archiving the near-real-time OGC catalogue service.

3. Methodology

3.1. Near-Real-Time OGC CSW Server and Metadata Repository

There are several OGC CSW-compatible server implementations in the Open Source Geospatial Foundation (OSGeo) community, as listed in Table 1. Among them, GeoNetwork is the earliest implementation written in Java as well as a popular one. GeoNetwork provides powerful metadata editing and search functions as well as an interactive web map viewer. pycsw is the only OGC CSW implementation written in Python. The advantages of pycsw are the ease of deployment, ease of configuration, and its support of multiple metadata models, such as ISO 19115 geographic metdata, Dublin Core metadata, DIF (Directory Interchange Format) metadata, FGDC (Federal Geographic Data Committee) Metadata, etc. Compared with GeoNetwork, pycsw is a lightweight implementation, and it is easier to incorporate in other Python libraries, which provide more efficient ways to achieve the near-real-time OGC catalogue service. Thus, pycsw is recommended as an implementation of the near-real-time OGC CSW server. For the metadata repository that pycsw works with, a PostgreSQL database with PostGIS enabled is the first choice since pycsw can make use of PostGIS spatial functions and native geometry data types for better spatial query operations. In addition, Nginx can be coupled

with pycsw to enhance CSW server performance. Nginx is a very popular web server with high performance due to its lightweight quality and easy workability and extensibility.

Table 1. CSW server implementations.

Name	Programming Language	Year Started
GeoNetwork	Java	2001
GeoServer with CSW plug-in	Java	2006
pycsw	Python	2010
deegree	Java	2014

3.2. Near-Real-Time Catalogue Update Approaches

We propose two catalogue update approaches: event-based and timing-based updates, according to whether the data are owned or managed by the provider of the catalogue service. For the scenario in which the provider of the catalogue service owns or manages the data, the event-based update approach is recommended, as shown in Figure 2. The event-based update is a form of passive catalogue update mechanism. The catalogue update task does not run until it is notified. In other words, a notify message is required to be sent by data owners or managers. For the scenario in which the provider of a catalogue service does not own or manage the data, only the timing-based update approach is applicable, as shown in Figure 3. The timing-based update is a form of active catalogue update mechanism. The catalogue update task runs periodically at fixed times, dates, or intervals, whether the data sources have incurred changes or not.

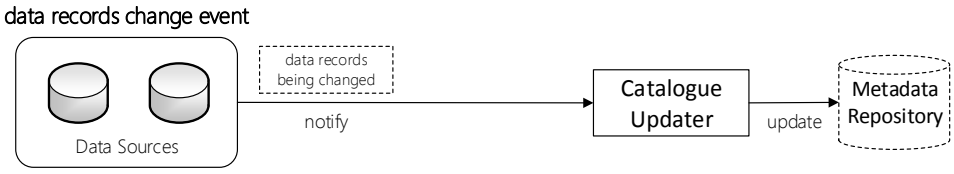


Figure 2. Event-based catalogue update.

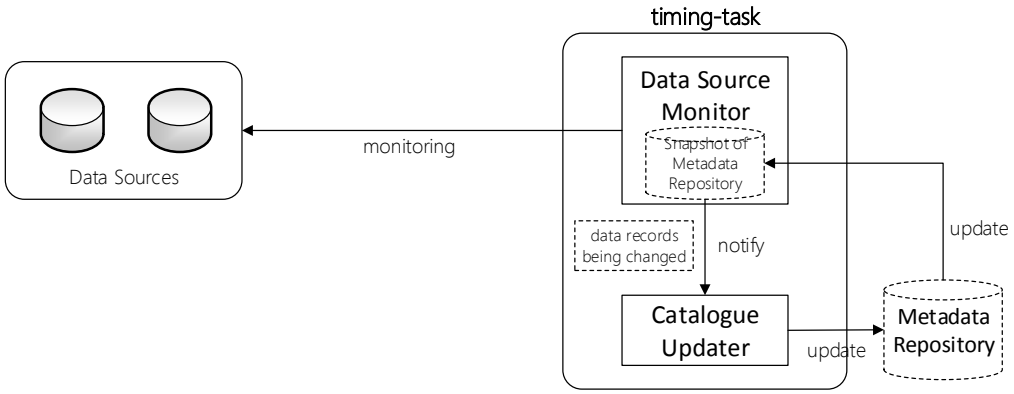


Figure 3. Timing-based catalogue update.

The event-based update is more accurate at the time of starting the catalogue update task, but it requires that data owners or managers notify the catalogue service providers, and this prerequisite is sometimes not easy to satisfy. The timing-based update approach is more widely applicable, even if it is likely to be less efficient. Therefore, we further focus on the timing-based update approach for the catalogue repository described in this study.

3.3. Timing-Based Catalogue Update Implementation in Parallel

As shown in Figure 3, the timing-based catalogue update is commonly implemented based on a time-based job scheduler. Most operating systems (OSs), such as Unix and Windows, provide OS-level job scheduling capabilities. Many programs can achieve relevant job-scheduling capabilities as well. Since data sources do not notify the catalogue updater, the data source monitor is proposed for tracking data sources and discovering data record changes. Since the metadata repository is normally built on a regular database, frequently creating connections and being connected to the database for the catalogue update will consume more database connection resources, and will take more time. Thus, instead of directly using the database of the metadata repository, a snapshot of the metadata repository is designed and used by the data source monitor.

3.3.1. Data Source Monitor

The data source monitor is implemented based on the comparative analysis between the data source and the metadata repository. We first investigated several primary data portals (USGS EarthExplorer, NASA EOSDIS, Unidata, GCMD, etc.) to understand how the data are archived and catalogued. Based on our investigation, the proposed data-archiving model is shown in Figure 4.

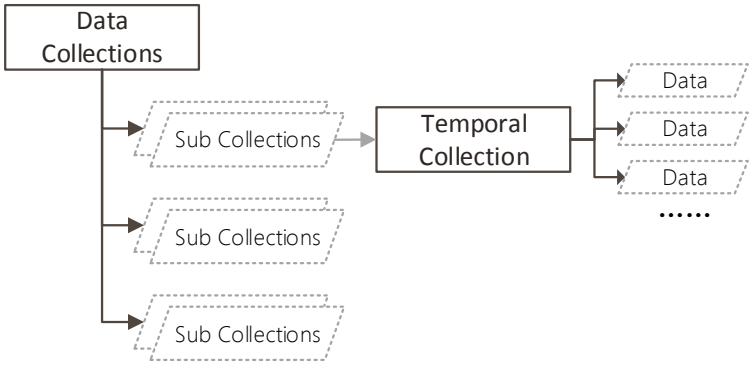


Figure 4. Data-archiving model.

Figure 4 shows that the data are usually archived at two levels: collection level and granule level. The collections can have sub-collections, and they are a form of hierarchical structure. Taking remote-sensing image data as an example, the data with the same sensor type can be one type of collection, the data with the same tile can be another type of collection, and the data with the same date can be another type of collection, etc. The granule level refers to the actual data, which are accessible via a downloadable data file or web services. Since the amount of geoscience data is normally very large, data collection is very useful in cataloging and archiving data. In particular, for the near-real-time data, one important type of collection for which the data will normally be archived is the temporal collection type; that is, collections identified with dates or times. Therefore, the data source monitor does the comparative analysis on the temporal collection type. The algorithm of comparative analysis is implemented based on the temporal topology relationship. With the temporal collection type, the data source monitor can quickly find the changes in data records and improve the efficiency of the catalogue update

The snapshot of the metadata repository is designed based on the data-archiving model, and it does not copy all metadata information of the repository. The elements of the snapshot track the elements of the data source monitor, including the URL of multi-level collections, the date and time of the temporal collection, and the identifier and the URL of the data. The snapshot adopts a node-key-value structure, as shown in Figure 5. The URL path of hierarchy collections is the node of the snapshot, which facilitates parallel processing on different servers. The date of a temporal collection can be the key of the snapshot, and sometimes the time of the temporal collection can be the key as well.

The key is used for comparison with the data sources in the comparative analysis algorithm. The value part of the snapshot is a list of the data corresponding to the temporal collection. Each element of the list only records the data identifier and its accessible URL. More metadata information is accessed through the URL in the update phase, not the monitoring phase. This idea also helps to save much more time in monitoring the data record changes, and the data list in the value node can be processed in parallel at the thread level on a single server.

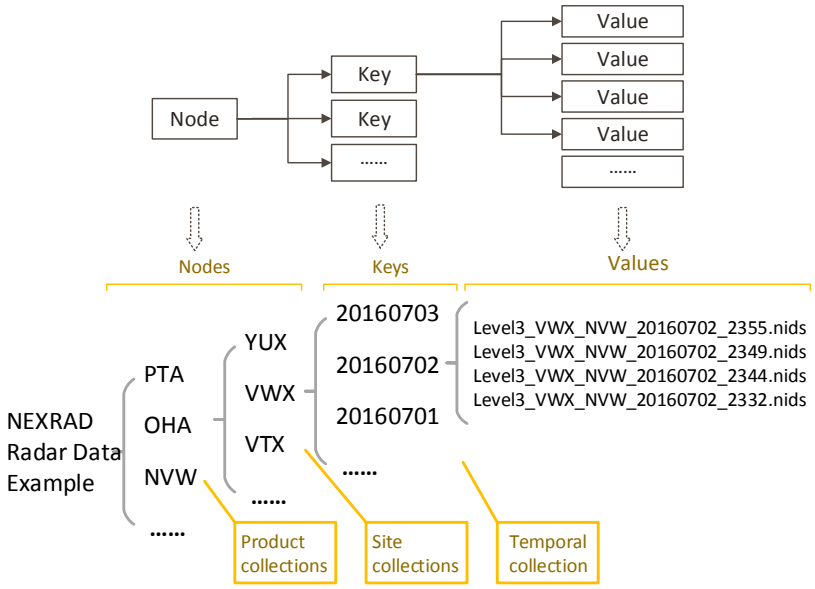


Figure 5. Snapshot file structure.

The data source monitor is implemented based on Pandas, a popular package in the Python community. Pandas provides high-performance, easy-to-use data structures, and data-analysis tools. Pandas has three fundamental data structures: Series, DataFrame, and Panel, representing a one-, two-, and three-dimensional labeled array, respectively. The axis labels are collectively referred to as the index. In addition, with Pandas, built-in functions and operators can be directly used in the comparative analysis algorithm based on the temporal topology relationship. In addition, Pandas provides a few I/O options, including CSV, TXT, JSON, HTML, HDF5, Excel, etc. We use the DataFrame structure of Pandas with HDFStore to implement the snapshot. The snapshot file is stored in HDF5 format.

3.3.2. Cluster-Based Catalogue Update in Parallel

When an extremely large amount of data needs to be updated in a short time, a catalogue update on a single server machine is usually not qualified for near-real-time processing. Therefore, we propose a cluster-based parallel processing approach. The cluster is composed of several server machines. They work together to execute a large task and have more powerful capabilities than a single server machine in terms of computation, storage, and scalability.

There are three primary parallel process frameworks: MPI (Message Passing Interface), OpenMP (Open Multi-Processing), and MapReduce (Hadoop). Both MPI and Hadoop can be run on a cluster, while OpenMP cannot be because of its shared-memory multiprocessing. In order to utilize clusters to accelerate catalogue updates, we employed a Python MPI framework, called mpi4py, to implement catalogue updates in parallel. The principle of MPI is that a serial of MPI processes is started and they execute the same code. Each process has an identifier (called rank in MPI) to mark what the process is. It should be noted that the variants in the code are process specific, even though they have the same variant name. Several mechanisms of communications are provided for message passing between multiple processes in mpi4py. They are point-to-point (sends, receives) and collective

(broadcasts, scatters, gathers) communications of any pickable Python object (pickable refers to the Python object is able to be converted into a byte stram), as well as optimized communications of a Python object exposing the single-segment buffer interface. Point-to-point communication enables the transmission of data between a pair of processes, one side sending, the other receiving; such collective communication allows the transmission of data between multiple processes of a group simultaneously.

In the process of a catalogue update, collective communication is adopted to split a large catalogue update task, because collective communication endeavors to utilize all the time of all processes for data transmission between multiple processes. It can be seen that collective communication is more efficient when applying a catalogue update. Based on the snapshot file for data monitoring, a large catalogue update task is scattered in clusters in terms of nodes of the snapshot. The nodes store data collections, as shown in Figure 6. With cluster-based parallel processing, efficiency gains are in proportion to the number of server machines in the cluster.

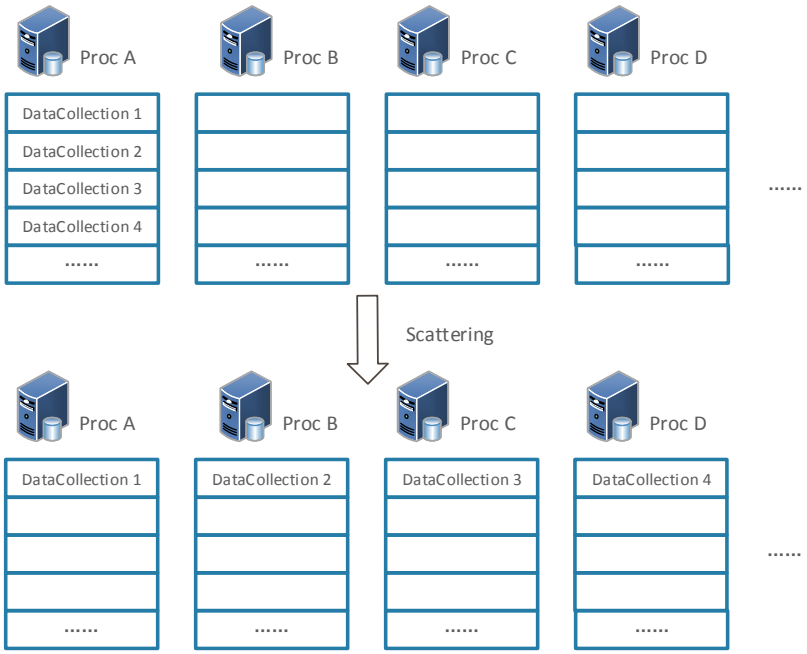


Figure 6. Cluster-based catalogue update.

4. Application and Catalogue Update Efficiency Analysis

In this study, Unidata data are taken as the application case. Unidata is a diverse community that shares geoscience data and software tools and that supports the enhancement of Earth-system education and research [13]. It is a member of the University Corporation for Atmospheric Research (UCAR) Community Programs, funded by the U.S. National Science Foundation. Unidata helps researchers and educators acquire and use Earth-related data, most being either real-time or near-real-time data. The real-time data are available online via the Unidata Internet Data Distribution (IDD) system. The Unidata IDD is an event-driven network of cooperating Unidata Local Data Manager (LDM) servers that distributes Earth science data products in near real time over wide-area networks [18]. The Unidata IDD is a primary source of real-time atmospheric science data and it has expanded from the initial U.S.-centric system to one that includes a few countries outside the United States [14].

4.1. Unidata IDD Data and Catalogue

Unidata IDD data involve forecast model output data, satellite data, radar data, and observation data, as shown in Table 2. The satellite data mainly involve GOES data, which includes visible, infrared,

water vapor, and sounder images. The radar data involve Terminal Doppler Weather Radar (TDWR) Level III and NEXRAD (Next Generation Radar) WSR-88D radar Level II and III products. As can be seen in Table 2, the satellite and radar data comprise extremely large volumes and are updated at a high frequency. This provides a good case for demonstrating a near-real-time OGC catalogue service for geoscience big data.

Table 2. Unidata IDD data and update frequency.

Dataset Category	Update Frequency	Count
Satellite data		
NESDIS GOES satellite data	5 min	140,000
Radar data		
NEXRAD Level II radar WSR-88D	3–10 min	660,000
NEXRAD Level III radar products		44,900,000
TDWR Level III radar		3,200,000
NCEP and FNMOC Forecast Model		3900
Rapid Refresh (RAP) High Resolution Rapid Refresh (HRRR)	1 h	1300
Short Range Ensemble Forecasting (SREF) Wave Watch III (WW3)	3 h	800
Global Forecast System (GFS) North American Model (NAM) Global Ensemble Forecasting System (GEFS)	6 h	1400
Downscaled GFS (DGEX)	12 h	60
NAVY Global Environmental Model (NAVGEM) Forecast of Aerosol Radiative Optical Properties (FAROP)	6 h	120
Coupled Ocean/Atmosphere Mesoscale Prediction System Navy Coupled Ocean Data Assimilation (NCODA) Model Wave Watch 3 (WW3) Model	12 h	230
NCEP analyses data		6100
Multi-Radar Multi-Sensor (MRMS) analysis Real Time Mesoscale Analysis (RTMA)	1 h	5000
National Digital Forecast Database (NDFD)	30 min	1100
Observation data		270
METAR Station data Surface buoy point data Surface synoptic point data	24 h	270

All these data are currently served through the THREDDS (THematic Real-time Environmental Distributed Data Services) server system, which is developed and is operated by Unidata. THREDDS provides information about the availability of datasets and the services and protocols to access them [19,20]. The protocols and services supported in THREDDS include OPeNDAP, Open Geospatial Consortium (OGC) WMS and WCS, HTTP, NetCDF Subset Service (NCSS), and ncISO Services [21]. THREDDS also has a hierarchical dataset catalogue, which provides virtual directories of available data and their associated metadata, but does not provide a data query interface. Thus, the OGC catalogue service is required to facilitate the discovery and interoperability of Unidata IDD data. The THREDDS catalogue structure is a form of hierarchical directory, which is applicable to the methodology proposed in Section 3.

4.2. Prototype of Near-Real-Time OGC CSW Service for Unidata IDD Data

A prototypical near real-time OGC CSW service for Unidata IDD data has been developed as a part of the CyberConnector project funded by the EarthCube program of the U.S. National Science Foundation (NSF). CyberConnector is designed to bridge sensors and Earth science models by extensively adopting open geospatial standards/specifications, such as the OGC Web Processing Service (WPS), Sensor Planning Service (SPS), Web Coverage Service (WCS), and Catalogue Service for the Web (CSW). It automatically prepares and customizes both historic and near-real-time Earth observation data and on-demand derived products, based on requirements of Earth science models, and feeds the prepared data into the models. Unidata is one of the important near-real-time data sources.

The framework of the prototype is shown in Figure 7. We employ five server machines, which work as a cluster, in this application case. One machine is responsible for providing OGC CSW services. The other four machines are responsible for monitoring real-time data sources and updating the metadata repository frequently. The prototype runs on the Ubuntu operating system and is built in the Python environment. pycsw, pandas, and mpi4py are all Python packages. "pycsw" is served for providing OGC CSW services, "Pandas" for data monitoring analysis, and "mpi4py" for parallel catalogue updates. An OS-level job scheduler "cron" is used for timing-based catalogue updates. The cluster-based catalogue update program in Figure 6 is scheduled with the Linux job scheduler "cron." The cluster-based catalogue update program starts four processes for executing catalogue updates in parallel. We also developed a THREDDS catalogue crawler for fetching ISO 19115 metadata from the THREDDS catalogue. In order to test the OGC CSW service, we developed a web CSW request builder, as shown in Figure 8a. With this builder, users can build any CSW request against Unidata IDD data and make a query with the request. Figure 8b shows the XML-encoding response of the near-real-time CSW service.

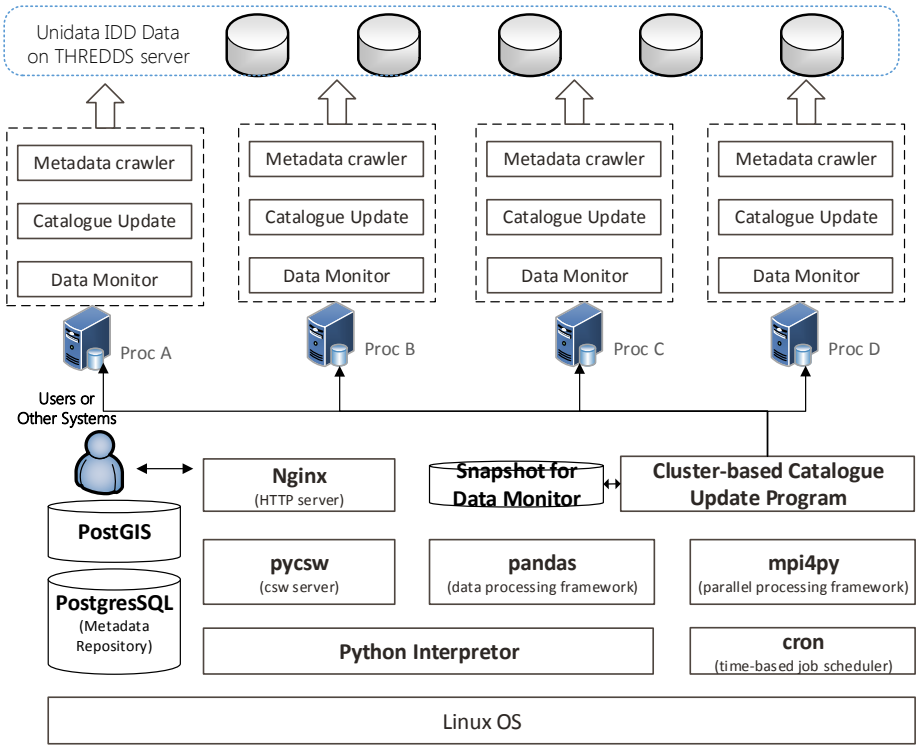
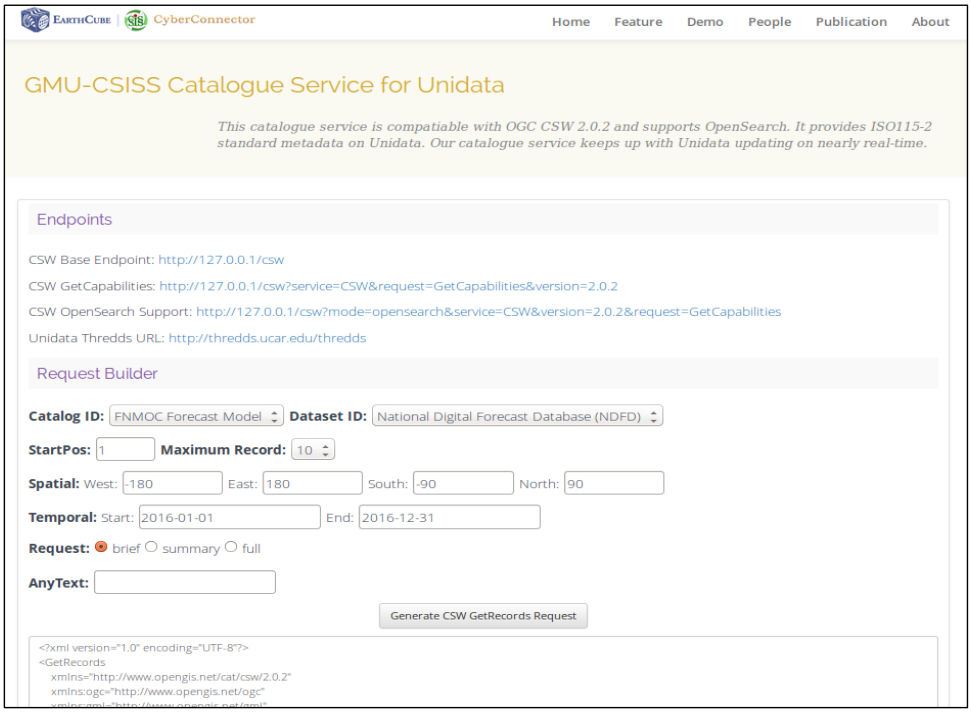


Figure 7. Framework of the near-real-time OGC CSW service for Unidata IDD data.



(**a**)

(**b**)

Figure 8. (**a**) CSW request builder for querying Unidata IDD data; (**b**) XML-encoding response of the CSW service.

4.3. Experiment and Result for Catalogue Update Task

Unidata IDD archives five categories of datasets, as shown in Table 2. In order to estimate the elapsed time to register them, we first carried out an experiment to calculate the average time to register one metadata record in our prototype system. The experiment ran on a single server with an Intel Xeon Processor E7 v3 (10 cores/20 threads), 24 GB RAM, and 16 TB storage. The network bandwidth was 90 Mbps. We chose some NEXRAD Level III radar data for testing, as shown in Table 3. With the average time of 1.38 s per record, we estimated the elapsed time to register all online metadata in terms of dataset categories, as shown in Table 4. We also estimated the elapsed time to register new online metadata later, as shown in Table 5.

Table 3. Elapsed time to registering NEXRAD III radar metadata for five sites.

Site Code	Begin Time	End Time	Elapsed Time (s)	Count	Elapsed Time to Register One Metadata Record (s)
PAKC	12:24:06	12:31:15	429	282	1.52
KTLX	14:48:05	14:53:37	332	262	1.27
KPBZ	15:59:32	16:03:02	213	160	1.33
KABR	17:05:26	17:08:07	161	141	1.14
FOP1	18:31:15	18:37:38	383	234	1.64
		Average:			1.38

Table 4. Estimated time to register all online metadata into CSW catalogue for the Unidata IDD data.

Dataset Category	Data Count	Estimated Time
Satellite data	140,000	54 h
NEXRAD Level II radar data	660,000	10.5 d
NEXRAD Level III radar products	44,900,000	717 d
TDWR Level III radar products	3,200,000	51 d
NCEP and FNMOC Forecast Model	3900	1 h, 30 min
NCEP analyses data	6100	2 h, 20 min
Observation data	270	6 min

Table 5. Estimated time to register new online metadata each time.

Dataset Category	Update Frequency	Update Count of New Data	Estimated Time
Satellite data	5 min	78	2 min
NEXRAD Level II radar data		166	4 min
NEXRAD Level III radar products	3–10 min	11232	4 h
TDWR Level III radar products		810	19 min
NCEP and FNMOC Forecast Model	1, 3, 6, 12 h	59	<2 min
NCEP analyses data	30, 60 min	13	<1 min
Observation data	24 h	3	5 s

The time in Table 4 can be used to estimate the elapsed time to initialize the catalogues, and the time in Table 5 can be used to estimate the elapsed time to update them. For the satellite data, catalogue initialization could take 54 h. It is acceptable to use a single task to process, but dividing it into multiple tasks is preferred. With the cluster used in this experiment, the time to initialize the satellite data was reduced to 13 h. For the radar data, it is impossible to process the NEXRAD III data with only two server machines because the radar data have an extremely large number of data items, reaching into the tens of millions. It requires a powerful cluster with high parallel processing capabilities for better efficiency. For other data catalogues (Forecast Model, NCEP analyses, and observation data), a single task is able to update them frequently.

Therefore, we set up the following tasks for the catalogue initialization and catalogue update phases, as shown in Table 6. Taking the satellite data as an example, the estimated time to initialize the catalogue was 54 h (Table 4); when we set up 10 parallel tasks, the catalogue initialization finished within 6 h. The actual elapsed time was 5 h and 42 min (Table 7), which is acceptable and meets the goal. We also set up two timing tasks in the catalogue update phase for the satellite data to complete the update in less than 2 min. Thus, the schedule is set up for every 2 min. The actual elapsed time was 1 min and 18 s (Table 7), which also meets the goal. All the elapsed times for initializing and updating catalogues in the experiment are listed in Table 7, which achieves the goal of a near-real-time OGC catalogue service.

Table 6. Tasks and schedules for monitoring Unidata IDD.

Dataset Category	Catalogue Initialization Phase	Catalogue Update Phase	Schedules
Satellite data	10 tasks	2 timing tasks	every 2 min
NEXRAD Level II radar data	20 tasks	4 timing tasks	every 2 min
NEXRAD Level III radar products	N/A	N/A	N/A
TDWR Level III radar products	30 tasks	20 timing tasks	every 4 min
NCEP And FNMOC Forecast Model	1 task	1 timing task	every 30 min
NCEP analyses	1 task	1 timing task	every 30 min
Observation data	1 task	1 timing task	every 6 h

Table 7. Actual elapsed times for catalogue initialization and update phases.

Dataset Category	Catalogue Initialization Phase	Catalogue Update Phase
Satellite data	5 h, 42 min	1 min, 18 s
NEXRAD Level II radar data	14 h	1min, 26 s
NEXRAD Level III radar products	N/A	N/A
TDWR Level III radar products	1 d, 18 h	2 min, 43 s
NCEP And FNMOC Forecast Model	1 h, 12 min	1 min, 55 s
NCEP analyses	2 h, 3 min	25 s
Observation data	5 min, 36 s	5 s

5. Discussion and Conclusions

This article discusses the big data issue in geoscience. A near-real-time OGC catalogue service focuses more on the velocity and volume characteristics of big data, which is different from regular OGC catalogue service studies. We proposed the framework of a real-time OGC catalogue service and presented a series of efficient catalogue update methods to ensure that the OGC catalogue is consistent with data sources. An application case of providing a near-real-time OGC catalogue service for Unidata IDD data was demonstrated based on the proposed methods and a prototypical near-real-time OGC catalogue service was developed. The results show that our methods and prototype are capable of dealing with ten million data items in near real time.

In addition, we conclude the following:

(1) Periodic catalogue updates are essential for providing a near-real-time OGC catalogue service for big data. Catalogue update frequency and the number of catalogue update tasks are important factors in designing a catalogue update schema, which involves how many update tasks are executed in parallel and how often the tasks are automatically executed.

(2) In this study, only one catalogue repository, which is normally based on one database, is used for the OGC catalogue service. However, it is better to combine multiple catalogue repositories to provide a near-real-time OGC catalogue service for big data. Thus, we hope OGC or other related communities may provide the schema for an OGC catalogue service with support for multiple catalogue repositories. This will make the catalogue more robust when dealing with big data issues.

(3) The methods proposed in this paper can be applied to most geoscience data sources because they all archive their data as hierarchical levels composed of classification collections, temporal collections, and datasets from top to bottom. The catalogue update methods presented herein are proposed based on these hierarchical archiving levels. Thus, the contribution of this study is not applicable to Unidata. Unidata is just one of the data sources that are qualified with the data-archiving structure.

(4) In particular, this study provides a solution for third-party organizations that do not own the data resources to implement a near-real-time OGC catalogue service. Providers of an OGC catalogue service that own the data resources may have particular methods for improving the efficiency of catalogue updating, but this is beyond the scope of the present study.

In the future, we plan to further study the approaches for improving time efficiency to deal with billions of data resources. Some popular techniques for handling big data, such as Spark and Hadoop, merit consideration when providing a near-real-time OGC catalogue service. A snapshot of a catalogue repository with better I/O performance could be another improvement, and multiple catalogue repositories with snapshot support will be more valuable in serving big data from multiple data sources.

Acknowledgments: This research was supported by a grant from the National Science Foundation EarthCube program (Grant No. ICER-1440294, Pi: Liping Di).

Author Contributions: Jia Song proposed, designed, and implemented the algorithm, performed the experiments, summarized the findings, and wrote the paper. Professor Liping Di conceived the research, reviewed, and modified the paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Nebert, D.; Whiteside, A.; Vretanos, P. OGC 07-006r1: Catalog Service Specification, Version 2.0.2. 2007. Available online: http://portal.opengeospatial.org/files/?artifact_id=20555 (accessed on 18 March 2016).
2. Bai, Y.; Di, L. Providing Access to Satellite Imagery through OGC Catalog Service Interfaces in Support of the Global Earth Observation System of Systems. *Comput. Geosci.* **2011**, *37*, 435–443. [CrossRef]
3. Bai, Y.; Di, L.; Chen, A.; Liu, Y.; Wei, Y. Towards a Geospatial Catalogue Federation Service. *Photogramm. Eng. Remote Sens.* **2007**, *73*, 699–708. [CrossRef]
4. Chen, A.; Di, L.; Wei, Y.; Bai, Y.; Liu, Y. An Optimized Grid-Based, OGC Standards-Compliant Collaborative Software System for Serving NASA Geospatial Data. In Proceedings of the 30th Annual IEEE/NASA Software Engineering Workshop, Columbia, MD, USA, 24–28 April 2006; pp. 159–166. [CrossRef]
5. Chen, Z.; Chen, N. Use of Service Middleware Based on ECHO with CSW for Discovery and Registry of MODIS Data. *Geo-Spat. Inf. Sci.* **2010**, *13*, 191–200. [CrossRef]
6. Di, L.; Yu, G.; Shao, Y.; Bai, Y.; Deng, M.; McDonald, K.R. Persistent WCS and CSW Services of GOES Data for GEOSS. *Geosci. Remote Sens. Symp. IEEE.* **2010**, *38*, 1699–1702.
7. Shao, Y.; Di, L.; Bai, Y.; Wang, H.; Yang, C. Federated Catalogue for Discovering Earth Observation Data. *Photogramm. Fernerkund. Geoinform. Jahrg.* **2013**, *2013*, 43–52. [CrossRef]
8. ECHO. What is ECHO? 2016. Available online: https://wiki.earthdata.nasa.gov/pages/viewpage.action?pageId=26543757 (accessed on 12 June 2016).
9. Battrick, B. The Global Earth Observation System of Systems (GEOSS) 10-Year Implementation Plan Reference Document. ESA (European Space Agency) Publications Division, 2005. Available online: http://www.earthobservations.org/documents/10-Year%20Implementation%20Plan.pdf (accessed on 12 June 2016).
10. Beyer, M.A.; Lapkin, A.; Gall, N.; Feinberg, D.; Sribar, V.T. 'Big Data' Is Only the Beginning of Extreme Information Management. 2011. Available online: http://www.gartner.com/id=1622715 (accessed on 26 July 2016).
11. Mauro, A.D.; Greco, M.; Grimaldi, M. A Formal definition of Big Data based on its essential Features. *Libr. Rev.* **2016**, *65*, 122–135. [CrossRef]
12. Laney, D. 3D Data Management: Controlling Data Volume, Velocity and Variety. 2001. Available online: http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf (accessed on 12 June 2016).
13. Unidata. Unidata Community. 2016. Available online: https://www.unidata.ucar.edu/community/index.html (accessed on 8 May 2016).
14. Yoksas, T.; Emmerson, S.; Chiswell, S.; Schmidt, M.; Stokes, J. The Unidata Internet Data Distribution (IDD) System: A Decade of Development. In Proceedings of the 22nd International Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology, Atlanta, CA, USA, 27 January–3 February 2006.
15. Martell, R. OGC 07-110r4: CSW-ebRIM Registry Service—Part 1: ebRIM Profile of CSW. 2009a. Available online: http://portal.opengeospatial.org/files/?artifact_id=31137 (accessed on 18 March 2016).

16. Martell, R. OGC 07-110r4: CSW-ebRIM Registry Service—Part 2: Basic Extention Package. 2009b. Available online: http://portal.opengeospatial.org/files/?artifact_id=31138 (accessed on 18 March 2016).
17. Voges, U.; Senkler, K. OpenGIS Catalogue Services Specification 2.0.2—ISO Metadata Application Profile. 2007. Available online: http://portal.opengeospatial.org/files/?artifact_id=21460 (accessed on 12 June 2016).
18. Yoksas, T.; Almeida, W.G.D.; Coelho, D.G.; Leon, V.C. Internet Data Distribution – Extending Real-Time Data Sharing throughout the Americas. *Adv. Geosci.* **2006**, *8*, 91–95. [CrossRef]
19. Domenico, B.; Caron, J.; Davis, E.; Kambic, R.; Nativi, S. Thematic Real-Time Environmental Distributed Data Services (THREDDS): Incorporating Interactive Analysis Tools into NSDL. *J. Digit. Inf.* **2006**, *2*.
20. Nogueira, R.; Cutrim, E.M. Extending THREDDS Middleware to Serve OGC Community. *Adv. Geosci.* **2006**, *8*, 57–62.
21. Bergamasco, A.; Benetazzo, A.; Carniel, S.; Falcieri, F.M.; Minuzzo, T.; Signell, R.P.; Sclavo, M. Knowledge Discovery in Large Model Datasets in the Marine Environment: The THREDDS Data Server Example. *Adv. Oceanogr. Limnol.* **2012**, *3*, 119–133. [CrossRef]