Searching for satellite data sets using Kepler, Metacat and EML

James Gallagher[1], Ben Leinfelder[2], Nathan Potter[1], Derik Barseghian[2]
1 OPeNDAP
2 NCEAS, University of California Santa Barbara
jgallagher@opendap.org, leinfelder@nceas.ucsb.edu, npotter@opendap.org, barseghian@nceas.ucsb.edu

***Abstract*—We present the results from building a data query module within the Kepler scientific workflow application. Our work focused on the query component of a larger use case from the Realtime Environment for Analytical Processing project where satellite-derived Sea Surface Temperature data were used to build match-up data sets as part of a workflow process. Kepler's integrated query capabilities allowed us to locate data described using the Ecological Metadata Language specification that was housed in a Metacat data catalog. Satellite data sets are significantly different from more traditional ecological data typically stored in Metacat, and while the resulting system worked well, it also highlighted areas where both Metacat and other satellite data-server software could be improved.**

Keywords—searching; workflow; integration; satellite data.

I. INTRODUCTION

Our motivation for this work was to determine how well suited the Kepler, Metacat and Ecological Metadata Language (EML) software components were for storing and querying descriptions of physical oceanography data. These data sets are often structurally very different from other traditional ecological data sets, although Sea Surface Temperature (SST) values are *fundamentally* ecological data. Kepler and Metacat have successfully provided effective storage and querying capabilities for ecological data, but we were curious to see how adaptable the spatial and temporal storage and search facilities provided by Metacat and EML would be to satellite data.

We found that many of the issues encountered were primarily due to the difficulty of building generic search systems for satellite data sets. The often-heterogeneous data storage schemes that are utilized by different data providers indicate a need for the server interface to abstract and generalize the details of any particular storage technique. While this can be accomplished using existing data servers for certain types of satellite data, to do so in the general case is difficult.

Providing effective query systems for satellite-derived data is important because these data sets are likely to become both more voluminous and more numerous. The 2007 National Academy of Sciences report on Environmental Data Management at NOAA estimated that satellite data volumes at NOAA alone will grow from ~3.5PB in 2007 to a projected level of over 40PB in 2020. It is very likely that current data organization patterns will persist and data discovery systems will face challenges similar to those documented here, but at a significantly larger scale. In fact, data management activities associated with storing and providing access to these data is considered "a significant data management challenge." [1]

A. The Relationship between Kepler, Metacat and EML

Kepler is open-source software that allows users to create scientific workflows, which are formal representations of the processes involved in scientific analyses. Kepler workflows may be used to connect a range of disparate software, and are saved in formats that are easily exchanged, re-run, versioned, and archived. [2] Metacat provides data set cataloging services to Kepler using Ecological Metadata Language (EML) documents. The EML specification [3] includes four basic element types; the *dataset* element is used to describe "broad information such as the title, abstract, keywords, contacts, maintenance history, purpose, and distribution of the data themselves." EML also contains elements for specifying spatial and temporal coverage of the data and supports grouping of related data objects into a single aggregated data set.

Metacat is designed to store, index and query any XML document, but is tailored for dealing with EML. Kepler's search system is able to quickly find EML-described data using predefined sets of indexed fields, including the use of temporal and spatial constraints. The Kepler workflow system interacts with Metacat using EarthGrid (formerly "EcoGrid") web services (see Fig. 1). The EarthGrid connects a number of independent systems and networks, providing access to data and metadata stored at distributed nodes. [4]

It would seem that given the Metacat and EML features for spatial and temporal data, they would be well suited for storing and querying metadata that describes satellite-derived data sets. However, this repurposing posed new development challenges.

B. The REAP Project and the Ocean use-case

The Realtime Environment for Analytical Processing (REAP) project consists (in part) of two very different use-cases which both use the Kepler scientific workflow system. These use-cases were specifically chosen to highlight fundamental assumptions inherent in the design of Kepler and to explore different solutions to the issues presented by these use-cases.

As different solutions were examined, we chose those that directly addressed the needs of the specific use-case but that also could be reused more generally. In this paper we will discuss implementing search features for the *Ocean use-case*.

The Ocean use-case entails building match-up[1] data sets for comparing different Sea Surface Temperature (SST) data sets. While a description of complete use-case is beyond the scope of this paper (see [5] and [6]), it is important to note that the data sets in this use-case are very different from those used in a typical ecology scenario - the subject of REAP's *Terrestrial Ecology* use-case [5]. The SST data sets used by the Ocean use-case are composed of thousands of individual files (e.g., a single Group for High-Resolution Sea Surface Temperature (GHRSST) data set at NOAA contains on the order of 10,000 files), each one holding SST values from one pass of a satellite. Roughly speaking, SST data sets may contain information in typical cartographic map projections (e.g., Lambert conformal) or they may contain data in *satellite coordinates* (i.e., each scan line is another increment along the satellite ground track) [7]. In this work we focused solely on those SST data sets that used a cartographic map projection where each pass contains data for the same geographic location. For all of the data sets used here, each file contains data corresponding to one satellite pass in the loose sense that while each scan line is actually a separate temporal event, they are clearly distinct from the scan lines captured by other passes of the satellite over the same geographic area. All the data files considered here are stored in 'self-describing' formats such as HDF4 or NetCDF that contain both data and metadata. Collections of these files are typically grouped into aggregate data sets, even though each file can also be considered a data set.

To satisfy the use-case, a user must be able to search for SST data sets that match spatial, temporal, and resolution parameters. Matches can then be input into the match-up data set workflow in Kepler. Difficulty arose when we attempted to implement this solution using the existing tools available within Kepler. As we will show, the issues are not specific to the implementation of Kelper but instead arise from fundamental data organization and representation paradigms used by our chosen data cataloging and search system.

C. *A bit more about the workflow*

In the Ocean use-case workflow, a user searches for a suitable SST data set to feed into the processing pipeline. The user must be able to search for data sets that intersect a region of interest specified by latitude, longitude, and time. In addition they must be able to narrow the search using both image resolution and parameter type. From a list of candidate data sets, one is chosen and used as input to the data processing pipeline. The data processing software is composed of a set of legacy software, written in Fortran, that reads data from a sequence of satellite images.

Figure 1. Data flow from the OPeNDAP server to Metacat/EarthGrid and Kepler. This is the data flow for information from the servers to the search system to the search client. OPeNDAP servers provide a hierarchy of THREDDS catalogs that describe all of the granules served. The THREDDS catalogs are crawled and the resulting granules are aggregated, resulting in EML documents. The EML documents are then stored in MetaCat which provides an API that Kepler uses to perform searches that return EML documents.

The data sets targeted by this use-case are large (greater than 10 GB) and are stored at a variety of government and university research laboratories where they are accessible using remote data servers. Because the data are staged at many different remote locations, it is important to be able search for them through a unified catalog system.

The SST data in this use-case are accessed using servers that implement the Data Access Protocol (DAP) developed as part of the Distributed Oceanographic Data System (DODS) and now extended and maintained by OPeNDAP [8]. The DAP provides a way to access remote data over HTTP and enables clients to request subsets of data using a *constraint expression* [8]. Using constraint expressions can both reduce data transfer sizes and relieve clients of performing subsetting tasks. Most DAP servers are used with data sets that are stored in files or groups of files (as is the situation for this use-case) and provide a discrete URL for each file. The URL is used to access the data in the file; each access can be made using a constraint expression; and each URL can provide metadata about the data contained in the file. DAP servers provide an additional service to clients: they shield them from having to know about the actual storage format of the data. The servers translate the data into the DAP data model for transport, so all data is sent over the network using the same representation regardless of the data set's native storage format.

[1] The term *match-up* refers to data sets that provide the same measured parameter for the same geospatial location (or, more generally, set of locations) using different sensors. For example, a match-up data set might consist of SST values from satellite data and SST in situ measurements from bouys.

II. THE SEARCH INTERFACE

The search interface was implemented as an *actor* in the Kepler workflow system. Kepler uses the term *actor* to denote any workflow component and to separate the components of a workflow from the overall workflow orchestration. There are many different kinds of actors including ones aimed particularly at scientific applications: remote data and metadata access, data transformations, data analysis, interfacing with legacy applications, Web service invocation and deployment, and provenance tracking. Once dragged to the workspace, the Advanced Search actor we developed automatically displays a dialog used to enter the search criteria (see Fig. 2). When the user clicks *Search* the dialog will be replaced with a list of data sets that match the specified criteria (see Fig. 3) and the user may choose one. The output of the actor is a list of URLs. This list of result URLs is then routed to the processing software as part of the workflow (examples of workflows can be seen in [5]).

The searching system relies on EML records stored in Metacat and accessed using the EarthGrid web services (see Fig. 1). The EML records are built and inserted into Metacat using a data server crawler that reads metadata from a predefined set of DAP servers and, using a simple rule-based system, builds EML documents describing the data sets it finds. A complete description of this crawler/aggregator software is beyond the scope of this paper but one important feature is that it identifies common patterns of multi-file satellite data sets and builds aggregations for them using EML. It does this by examining large collections of URLs collected from a site and grouping subsets of those URLs using patterns. Thus groupings (i.e., aggregations) of the URLs can be formed without the data provider making them explicit. The aggregator component of the software then encodes these aggregations using EML so that its output easily integrates into the MetaCat/EarthGrid/Kepler system.

The structure of the EML records used by the query system is shown in Fig. 4. As discussed previously, the EML data set element holds information about the aggregation, while information about each file that makes up the aggregation is held in an otherEntity element. In Fig. 4 the structure of the physical child element of otherEntity is shown. This is the element actually used to bind the URL that references a single file with a specific date and time.

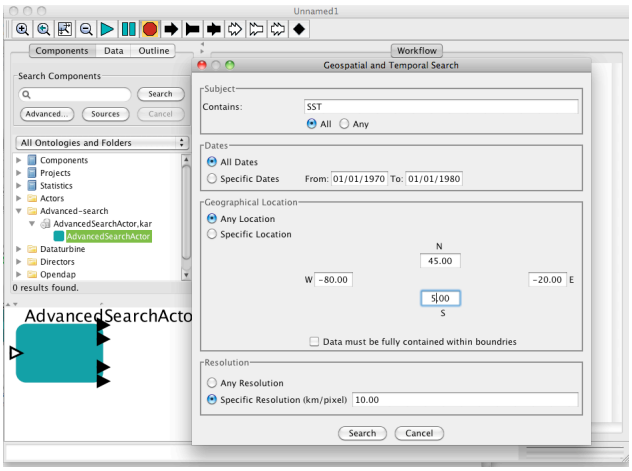

Figure 2. The search system interface implemented as an actor in the Kepler workflow system. Results from the search can be reviewed in a second pane and then fed into subsequent stages of a workflow (not shown).

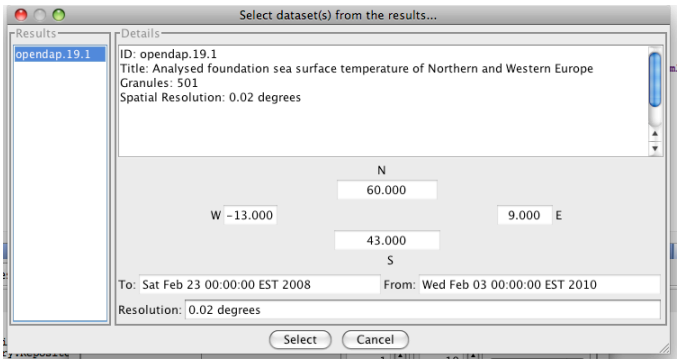

Figure 3. The result dialog. Users choose a single data set from the list of matches (here only one match was returned–shown in the left pane), browse metadata matching the query parameters and feed the result into the workflow.

```
<dataset/>
        <title/>
        <otherEntity/>
                <physical/>
                        <objectName/>
                        <encodingMethod/>
                        <url/>
                        <coverage/>
                                <temporal/>
                                <spatial/>
        ... (more otherEntity elements)
```

Figure 4. Within the *otherEntity* element, information about a single file is held in a *physical* element that contains a number of other child elements. This is repeated for each file in the data set. Spatial and temporal information is held in the *coverage* element.

III. DISCUSSION

One of the most important issues we confronted when building the search actor was how to handle the multi-file nature of the satellite data sets that play a central role in the Ocean use-case. Below we review the three approaches we considered and compare their merits and weaknesses. We found that while Metacat was generally flexible and extensible, there were

certain features of the system that required us to implement additional client side query refinements.

A. *Building aggregations for multi-file satellite imagery*

We examined three ways to form aggregations of the satellite data sets in this use-case:

- Building aggregations using the search system
- Building aggregations using the data servers
- Extending Metacat to better support aggregations

B. *Building aggregations using the search system*

Using EML documents to hold the aggregations provides a solution with a number of trade-offs. It de-couples the grouping of data URLs from the server, so the search system is no longer dependent on data providers building server-side aggregations. In this particular use-case, even though the technology for building and serving aggregations was available, it was not installed for most of the data sets. Even if the technology is installed, it may not be fully used, particularly by smaller laboratories, since it does require effort to configure and maintain. Moving control of the aggregations to the search system makes it easier to ensure uniformity among the data sets retrieved.

Building aggregations within the search system, however, has a number of drawbacks. First, the distributed nature of the system is subverted in that content generation cannot be spread among many different people and organizations. Building aggregations in the search system requires curation and aggregation be performed by the maintainers of that system. The complexity of the data sets compounds the problem; local experts may have knowledge about the data that the maintainers of the search system do not. There may be a considerable qualitative advantage to having the data provider build an aggregation. Another drawback is that some of the data abstraction capabilities a data server typically provides are lost. By building EML records that explicitly enumerate each URL in an aggregated data set, we are encumbering the client (the search interface in this case) with the task of selecting which of those URLs satisfies the search criteria.

An alternative approach to using single EML documents to hold the aggregations is to have Metacat store a single EML document for each URL in every data set. Using this scheme, the search interface would return all of the URLs that match the search criteria and the search interface would be responsible for forming the aggregations. If the aggregation step were skipped, one might assume that all returned imagery perfectly matched the search criteria. But a database with records for many SST data sets will likely return mixed records from different data sets that share the same space, time, and parameter values with similar resolutions but, for example, that differ in the specific algorithms used to compute the SST values. The user of such a system would be left to sort through tens or hundreds of thousands of URLs – effectively they would have to form the groups 'by hand,' an almost impossible task given the number of discrete items involved. Thus the search criteria used by the interface are necessary but not sufficient to select specific URLs for input in to the workflow in the general case.

We still could have adopted the *one EML document for each URL* scheme by building more intelligence into the search interface itself. When the interface received what would likely be 10,000 or more EML documents as the result of its query, it could have grouped those using other metadata in the documents. For example, it could have used the data set[2] title and the host name in the URL to form groups that would likely be correct.[3] However, doing this presents no real advantage over the case where a single EML document stores all of the URLs. The search client still must understand that the results of a query should be grouped before they can be used (so information-hiding is lost) and the task of forming the aggregations is moved away from the data sources to the search system (distribution of responsibility is lost). Building the aggregation capability into the search interface has one additional drawback. If the software that forms those groupings is found to have a flaw, it will have to be fixed in a subsequent release. However, a flaw in the EML stored in the Metacat database can be fixed by editing the EML document.

C. *Building aggregations using the data servers*

Using data server aggregations, a collection of two-dimensional 'granules' where the granules vary only in time can be combined to form a single three-dimensional data set. The DAP subsetting feature can then be used to access a latitude/longitude/time subset from this larger three-dimensional data set. That is, the data access operation performs the temporal search and subsetting operations. Effectively, that part of the search problem has been factored out of the search system and moved into the software that reads the data.

Building aggregations using the data servers is a technique that presents several significant advantages. The search system can store compact records that describe each aggregated data set, eliminating the coupling between the internal organization of the data sets and the search system. At the same time, this approach frees the search system's database maintainer from having to form the aggregations. Users of the workflow can be confident that the aggregations represented by the system are valid because the people closest to, and knowledgeable about, the data have built them.

---

[2] Note that when using DAP servers; each file is considered a unique data set. The aggregations are logical groupings that are imposed on the discrete elements. Even forming an aggregation using a DAP server does not mean the individual URLs are not also accessible.

[3] Another solution is to include the equivalent of a foreign key in the EML documents so that the search interface can know which are related. This is really only an incremental improvement, however, since the search client still has to know how to use the key (information hiding is lost).

Unfortunately support for server-side aggregation is far from universal, and, in fact, it could not be applied to any of the SST data sets used by this use-case. Even if server-side aggregations were available for the data sets in this specific use-case, relying on them would violate our goal of generality. Because we assumed, for the sake of simplicity, that the use-case would handle only data sets with uniform cartographic projections covering the same geographic area, we eliminated a significant number of potential data sources that only provided data with satellite coordinates. We would like to use data stored in satellite coordinates [7], but the individual files in these data sets cannot be aggregated using the simple techniques applied to cartographically and geographically uniform data. Furthermore, while the technological capability might be present to form server-side aggregations, it does not mean that every data provider will use it. Thus, a more general solution must address the case where a data set is available only by individually accessing each of its files (i.e., URLs) directly.

D. *Extending Metacat to better support aggregations*

At first glance, Metacat and EML provide a robust feature set for addressing the problems presented by the REAP Ocean use-case's searching component. EML can represent aggregations if and when the origin data server cannot provide that capability; Metacat can perform geospatial queries and, for this work, was extended to support temporal search criteria as well.

One limitation with the Metacat/EarthGrid system is the difficulty in processing very large EML files and/or returning very large numbers of EML elements as responses. While the response structure (and EML document structure) are well suited to tens or hundreds of records, with satellite data sets there are often thousands, and sometimes millions, of 'records' returned as part of a single query. This difference, many orders of magnitude in size, placed a strain on the components of the Metacat/EarthGrid system and required that that we build the custom search interface. While it may be impossible to completely address these scalability issues, there may be ways to mitigate them.

Metacat could be extended in two ways that would address these scalability problems and make it a more flexible tool for this kind of search interface. While Metacat queries can be constrained so as to return a subset of the element *type* in a document, it cannot form a subset of individual *instances* of those element types. Thus returns *all* of the requested elements in an EML document when *any* of those elements match the search criteria. This means that when the search interface is used to query a limited time range and finds an EML document that contains a single match, it will return all of the URLs for that data set, not just the ones that fell within the query's time range. If Metacat were modified to return only those elements that matched a parametric query (such as those *physical* elements that contain *coverage* elements which fall within a certain time range) then the search interface could eliminate any processing of the returned set of URLs.

A second improvement to Metacat would be to preserve the EML element hierarchy in the response it returns. In the current implementation, Metacat 'flattens' the responses making it difficult for the search interface (the recipient of the response) to detect errors that result from missing data in the original EML document. We found that errors did appear in a small fraction of the automatically generated metadata. While it would be best to detect and correct those errors at the source, increasing the overall robustness of the search system would also help trap the cases that will inevitably slip by.

IV. CONCLUSION

We found that building a search interface for the REAP Ocean use-case that used Metacat/EML/Kepler software worked well. Although these SST data are ecological data in the strictest sense, satellite data sets possess different characteristics in larger scales than the ecological data for which Metacat was originally targeted. In spite of these differences, we were able to build a query system using these tools that was not significantly different from other similar interfaces built in the past [9].

While EML imposes no theoretical limits of the number of discrete data objects contained in aggregate data sets, we encountered practical obstacles when using Metacat to query satellite data aggregations containing on the order of 10^4 data objects. The inability to query and directly retrieve specific data object records from within the containing aggregation was the most problematic limitation. Providing support for selective query behavior in a future release of Metacat would eliminate the need for post-processing Metacat search results and would better server the REAP Ocean use-case.

In addition, we found features that would have simplified our task were present but underutilized in the software that serves these data. We will investigate ways to simplify the deployment of these data-server-based aggregation techniques and encourage their increased adoption by data providers.

ACKNOWLEDGEMENT

This work was sponsored by NSF grant #0619060. We also thank Dr. Peter Cornillon for help with the Ocean use-case requirements, data server crawling technique and information about satellite data set growth.

REFERENCES

[1] National Research Council of the National Academies, "Environmental Data Management at NOAA: Archiving, Stewardship, and Access," Committee on Archiving and Accessing Environmental and Geospatial Data at NOAA, National Research Council, ISBN: 0-309-11210-9, 2007, pp18-19. http://www.nap.edu/catalog/12017.html.

[2] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, and S. Mock, Kepler: "An Extensible System for Design and Execution of Scientific Workflows," 16th International Conference on Scientific and Statistical Database Management, 2004.

[3] "Ecological Metadata Language (EML) Specification," Version 2.1.0, http://knb.ecoinformatics.org/software/eml/eml-2.1.0/index.html.

[4] D. Pennington and W. Michener, "The EcoGrid and the Kepler Workflow System: A New Platform for Conducting Ecological Analyses," ESA Bulletin (Emerging Technologies), 86:169–176, 2005.

[5] D. Barseghian, I. Altintas, M.B. Jones, D. Crawl, N. Potter, J. Gallagher, P. Cornillon, M. Schildhauer, E.T. Borer, E.W. Seabloom, and P.R. Hosseini, "Workflows and extensions to the Kepler scientific workflow system to support environmental sensor data access and analysis", presented at Ecological Informatics, 2010, pp.42-50.

[6] P. Cornillon, D. Crawl, and I. Altintas, "A user based approach to comparing SST fields," Presented at the GHRSST IX Science Team Meeting, Perros Guirec, France, 2008. https://www.ghrsst.org/documents/q/category/ghrsst-science-team-meetings/ghrsst-ix-science-team-meeting/.

[7] F. Monaldo, "Primer on the Estimation of Sea Surface Temperature Using TeraScan Processing of NOAA AVHRR Satellite Data, Version 2.0, S1R–96M–03," The Johns Hopkins University Applied Physics Laboratory, October 22, 1997, p11.

[8] J. Gallagher, N. Potter, T. Sgouros, S. Hankin, and G. Flierl, "The Data Access Protocol – DAP 2.0," NASA ESE-RFC-004.1.1, 2007, http://www.esdswg.org/spg/rfc/ese-rfc-004.

[9] P. Cornillon, J. Chamberlain, and D. Holloway, "The system integrator in distributed data systems: The OPeNDAP Data Connector (ODC)," *in* Sinha, A.K., ed., Geoinformatics: Data to Knowledge: Geological Society of America Special Paper 397, 2006, pp.230-1. doi: 10.1130/2006.2397(18).