

# Oceans and Climate Digital Library Portal

Ian Cumming<sup>1</sup> and Glenn Hyland<sup>2,3</sup>

1. *Insight4 Pty. Ltd., Hobart, Tasmania, 7001*

2. *Tasmanian Partnership for Advanced Computing, University of Tasmania, Hobart, 7005*

3. *Australian Antarctic Division, Kingston, Tasmania, 7050*

## Abstract

We have developed a Digital Library Portal that provides an integrated web-based user interface to a wide range of online oceanographic and climate data sets. The portal provides a single point of entry to the data sets, and displays them in a common and easy to navigate format. Moreover, it provides a means of searching the meta-data and other characteristics of the data sets. A key component of the portal design is the *search crawler*, which periodically checks specified locations (local directories or external websites) for updated or modified data sets. The crawler updates the portal with the most current state of every data set, and allows new data sets to be discovered and automatically added to the system. The portal will facilitate the sharing of earth systems science data sets, and enable scientists to more efficiently manage their data holdings and data analysis work flows. These are precisely the outcomes required to stimulate e-Research.

## 1. Introduction

The Tasmanian Partnership for Advanced Computing (TPAC) has established a distributed library of oceanographic and climate data [1]. Some of the elements of this digital library are:

- Intergovernmental Panel on Climate Change (IPCC) coupled ocean-atmosphere general circulation model output;
- WOCE V3 Global Data resource (searchable using the web-based *WOCE3 Viewer* developed at TPAC);
- NCEP2 re-analysis data;
- ocean surface height data from altimetry;
- Antarctic Weather Stations data and other commonly used climatologies.

The library is distributed between a number of agencies: the University of Tasmania and CSIRO Marine Research in Hobart; the CSIRO HPSC facility and the Bureau of Meteorology Research Centre in Melbourne; and the Australian Partnership for Advanced Computing (APAC) National Facility in Canberra.

The library is built on the Open-source Project for a Network Data Access Protocol (OPeNDAP) [2], which allows self describing data (such as netCDF or HDF) to be interrogated and delivered seamlessly across the web. Users are able to directly access the data using any application that has been linked with the appropriate OPeNDAP libraries. Applications include web browsers, GrADS, Ferret, Excel and common programming languages such as Matlab, C, Java and Fortran.

Before the development of the portal, the digital library was accessed via the TPAC website [1]. The library front page provided a list of categories of data, with links to the actual data sets available on each agencies website. This method of accessing the data suffered from several limitations, namely:

- no ability to search on most datasets – OPeNDAP servers only allow users to browse data set directories, and examine the attributes of individual data files;
- no consistent graphical layout or user interface – agencies hosting the data sets constructed and

maintained website pages in accordance with their own guidelines;

- new data sets had to be manually added to website pages;
- no user authentication or access control.

These limitations were not only prohibitive to the ease of use and navigation of the digital data, but also towards the ability of TPAC to both maintain and add new data sets.

## 2. Digital Library Portal

We developed the Digital Library Portal to overcome the limitations of the current system. In particular, we designed the portal to provide consistent and powerful methods to search through data sets and the files available within data sets; to simplify the management of data sets; and to provide authentication and access control mechanisms for sensitive data.

The portal was implemented within the GridSphere portal framework - an open-source framework developed in Java [3]. An important design consideration was to ensure the underlying elements of the portal were compatible with the proposed grid and web-based technologies to be used by the APAC National Grid Facility, and GridSphere has been chosen as the standard framework for portal development under the National Grid program [4].

A schematic overview of the portal is shown in Figure 1. We have used a three tier model, comprising front-end portlets, a data API, and a search crawler. Each component can be replaced independently of the others, allowing for greater versatility should the system need to be modified, or additional features added. The separation of the data API as an independent component allows additional portlets or stand-alone applications to share digital library data through a consistent mechanism.

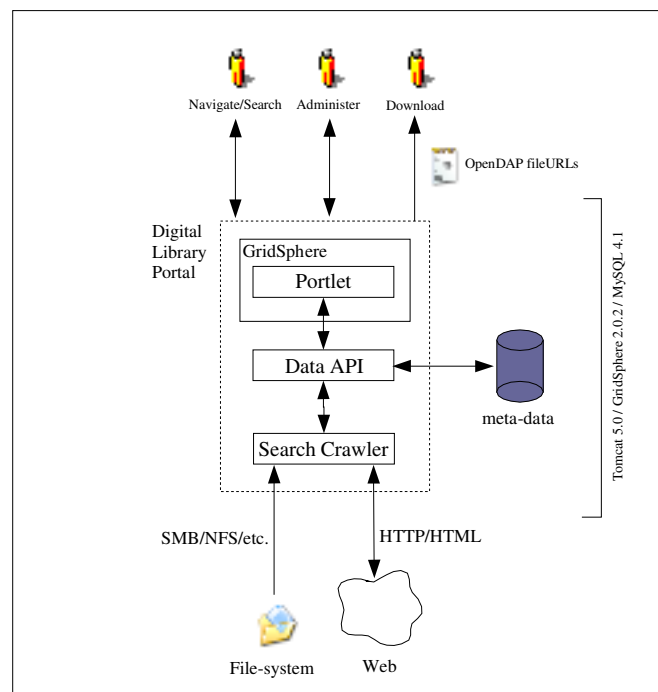


Figure 1: Digital Library Portal framework.

### 2.1 Data API

The data API is the main component of the portal architecture. It comprises a relational database and Java class interfaces, and provides an object-oriented interface to access data sets, files and their

associated meta-data. The API abstracts from the underlying relational structure of the database and provides the programmer with a rich set of classes and methods to store, search and retrieve data.

We have used a flexible data model that describes each discrete data element in an object-oriented fashion. The model is hierarchical, so that higher level objects encapsulate lower level objects in a one-to-many (1:N) relationship – shown conceptually in Figure 2. This approach allows the system to be represented as a rich object-oriented framework, allowing users to work with data concepts in the system as they would think about them naturally, and not as they are stored in the underlying database. The Data API is provided as a stand-alone Java library, and is independent from the implementation of other application components.

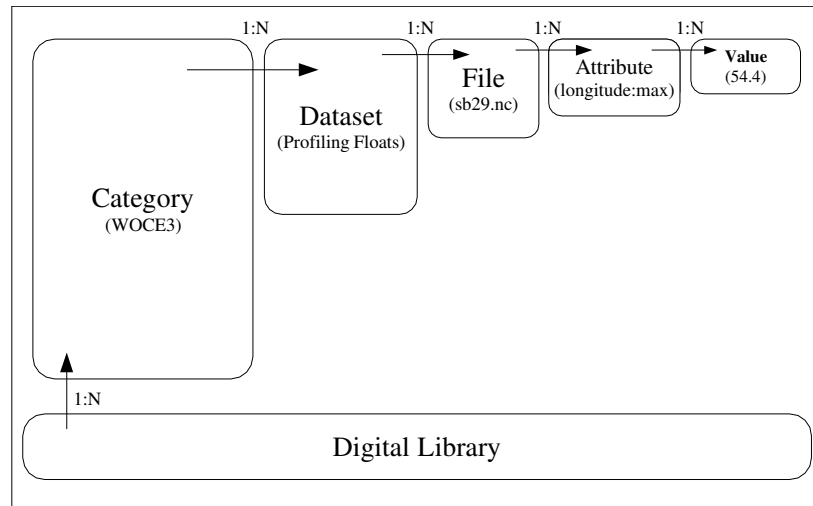


Figure 2: Overview of the digital library data model. Example labels are provided to illustrate the concept of this model.

## 2.2 Front-End

The front-end consists of a set of GridSphere portlets - a navigational portlet, an administrative portlet, and a help portlet. The portlets interact with data in the digital library through the stand-alone data API.

The GridSphere framework provides a choice for developers between two similar portal APIs: the GridSphere portlet API, (based on IBM WebSphere API); and the JSR 168 portlet API. The fundamental concepts of the portlet life-cycle, portlet modes, window states, sessions and request processing are almost identical between both APIs. They differ slightly in their concepts of configuration and customisation objects, as well as request/response objects [5]. We chose the GridSphere portlet API based on the availability of documentation and proven implementations of GridSphere applications at the time of design (August 2004). The differences between JSR 168 and our GridSphere implementation are superficial in terms of source-code. Support for JSR 168 would require only minor modifications to re-factor our application to this standard.

We also chose the GridSphere UI tag and visual bean library to construct the portlet interfaces as visual components. The library provides Java bean representations of the tags to be used in portlets for action handling or presentation logic. This greatly assisted the development of the user interface.

The navigational portlet consists of a website and a set of search/navigation forms that allow the user to specify either a file path or search criteria from which to select data sets. Figure 3 shows the

three main views of the portal front-end – the *main page*, which lists all available data set categories; the *data set view*, which lists all data sets within a particular category; and the *search view*, which is used for searching within data sets.

The user is able to collect links (OPeNDAP URLs) from various searches in a link collection (which behaves much like an online retailer's shopping cart). The links can be stored from many searches and later downloaded as a single file (Figure 4). This allows easy insertion of the data file references into scientific applications such as Matlab.

The administrative portlet provides the means of managing the data sets presented by the portal. It provides the functionality: to add and remove categories and data sets; to add and remove textual descriptions for categories and data sets; and to control what authentication is required to access specific data sets.

The front-end uses the data API to retrieve requested data for display. An instance of the data API is created for each user session viewing the portlet. As the data API instance is persistent between request/response calls to the portlet, only one database connection per user is required by the system, thereby lowering resource requirements. Similarly, subsequent requests for the same data objects can call upon the API's object cache, limiting database queries and improving display performance.

### **2.3 Search Crawler**

The search crawler is a console based Java application that populates the database with data set files and meta-data. Associated with each data set is a base URL. The crawler periodically checks these web sites, and updates the database entries for each data set using the files that are discovered.

The application works in a similar manner to a conventional search engine - starting from a specified base URL, and traversing hyper-links within a particular set of constraints. Specifically, the crawler searches web pages for OPeNDAP URLs. Using the Java OPeNDAP library [2], the crawler retrieves the meta-data from each file URL, and stores this using the data API. Navigational constraints such as restricting the file path and host name, constrain the crawler to specific regions of a web site, and therefore to one particular data set.

### **2.4 Database Performance**

Responsiveness and scalability are key requirements for ensuring usability of the portal. We have addressed these requirements by focussing on the performance of database operations in the data API. In particular, we optimised the underlying relational database structure, and applied strategies to reduce the in-memory row-comparisons on the database server. As a result, the data API is highly optimised for indexing and searching of file meta-data.

One core strategy we implemented was to assign each data set its own set of tables to store file attribute lists, attribute relationships and attribute data. This limits the database server to only scanning rows associated with the data set being searched. Additionally, this allows the database to scale without impeding the speed of querying other data sets.

To improve search queries and allow powerful filtering and sorting, each attribute is stored as a separate row in the underlying table. This allows the data attribute table to be joined back onto itself (using a JOIN query) enabling searching and sorting on multiple different attribute types. Therefore we are able to use a single optimised SQL statement for searching and sorting, which improves the

speed of queries because the application does not need to post-process any of the results.

To examine the performance gain from structuring the database on indexed tables, we compared the query time between indexed and non-indexed tables on a number of query operations. The queries were based on common user operations, such as browsing, searching, and sorting files based on a selected attribute.

Our test results are shown in Table 1. They show that indexed table queries provide exceptional performance gains against non-indexed table queries when sorting results. For select and search operations, indexed table queries performed faster than non-indexed queries, and were completed in less than 5 seconds – well within our bounds to ensure responsive interaction with the portal for these types of operations. These results quantify that optimising the database structure around row indexes is necessary to ensure usable operation of the portal when sorting results.

| <i>Query Type</i>              | <i>Indexed</i> | <i>Non-indexed</i> |
|--------------------------------|----------------|--------------------|
| Simple Select                  | 0.003s         | 0.605s             |
| Search (1 attribute)           | 0.112s         | 0.514s             |
| Search (2 attributes)          | 0.175s         | 0.964s             |
| Search (2 attributes) and Sort | 0.226s         | <b>190.260s</b>    |

Table 1: Query results for Indexed and Non-Indexed tables. The test data set was the WOCE3 sub-surface velocity data, and contained 257,036 table rows to store attribute data for 2,514 files. The results are the average query time of 3 trials.

## 2.5 Testing

Testing the portal on OPeNDAP sites with deep data hierarchies identified an issue related to those sites using on-line tape storage systems. Frequent and sequential searching on these sites can trigger bottlenecks on remote servers that may require access to tape-storage. In some cases this could result in denial of service to other users, or reduce the performance of other tasks on the remote server.

We addressed this problem by providing support for custom update indexes. The index is generated on the remote system periodically and contains a full indexed list of OPeNDAP file URLs, including last modified file dates. The crawler accesses this index instead of traversing the web site, and synchronises the database contents based on the file modification times provided.

This approach improves the performance of populating and updating the database using the crawler, but is limited to remote sites willing to provide an index file for this purpose.

## 3. Discussion

Anecdotally, access to information is the most significant obstacle to collaboration between researchers, clients and the wider community. Too often compromises are made to data and model results by the artificial requirements of bandwidth, local resources and the time it takes to obtain and read data. The development of the oceans and climate digital library portal will improve access to data, enhance collaboration, and thus reduce work-flow barriers. These are precisely the outcomes required to stimulate e-Research.

#### **4. Future Developments**

The APAC National Grid plans to transition to a more operational grid based on web service developments, which involves porting existing grid enabled projects into the web services environment [4]. An interesting avenue of use for the digital library portal is to provide data querying and access through an eXtensible Mark-up Language (XML) Web Service. The digital library does not currently provide any web based services of this type.

A web service would allow client-side applications and services to request data from the digital library over the web using XML. This would facilitate different uses and methods of access to the digital library other than through the portlet interface. For example, a client side application could use a web based service to provide the user with a simple file search query. This would allow the user to search for files without the need to log into the portlet interface.

Web Services for Remote Portlets (WSRP) is a standard from the OASIS group that defines a web services interface for accessing and interacting with interactive presentation-oriented web services [6]. Future enhancements to the digital library may include a WSRP layer, to allow additional services and applications to be built around the digital library data.

#### **References**

1. TPAC digital library main page  
<http://www.tpac.org.au/datasets/diglib.htm>
2. OPeNDAP home page  
<http://www.opendap.org>
3. GridSphere home page  
<http://www.gridsphere.org>
4. APAC Grid Program TWIKI page  
<http://www.vpac.org/twiki/bin/view/APACgrid/WebHome>
5. Comparing the JSR 168 Java portlet specification with the IBM portlet API:  
[http://www-128.ibm.com/developerworks/websphere/library/techarticles/0312\\_hepper/hepper.html](http://www-128.ibm.com/developerworks/websphere/library/techarticles/0312_hepper/hepper.html)
6. Organisation for the Advancement of Structured Information Standards (OASIS)  
<http://www.oasis-open.org/home/index.php>



(c)

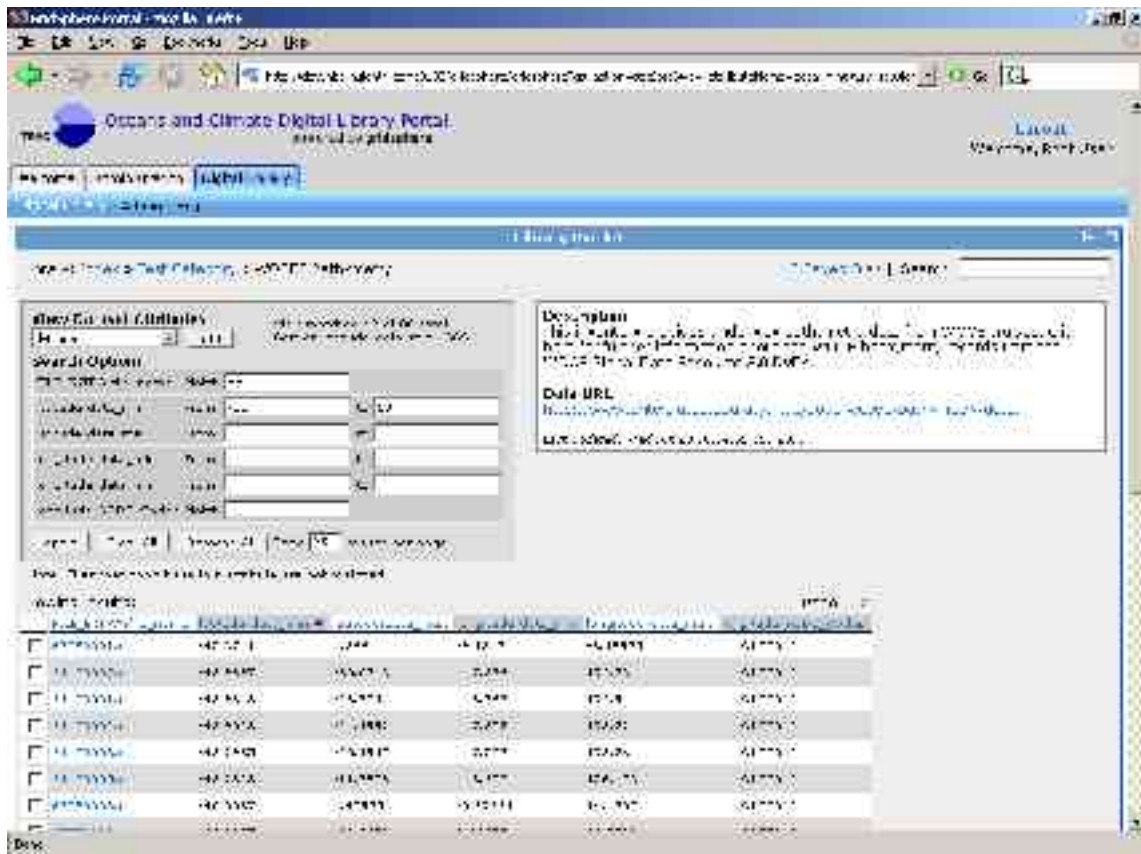


Figure 3: The front-end navigational portlets. (a) the *main page*, which lists all available data set categories; (b) the *data set view*, which lists all data sets within a particular category; and (c) the *search view*, which is used for searching within data sets.

