GADS: Using Web Services to access large data sets

Jon Blower[1*], Keith Haines[1], Chunlei Liu[1], Andrew Woolf[2],
Kecheng Liu[3], Adityarajsingh Santokhee[3], Ying Zhao[3]

[1] Environmental Systems Science Centre, Harry Pitt Building, University of Reading, Whiteknights, Reading, RG6 6AL
[2] CCLRC Rutherford Appleton Laboratory, Chilton, Didcot, Oxon, OX11 0QX
[3] Department of Computer Science, School of Systems Engineering, University of Reading, Whiteknights, Reading, RG6 6AY

[*] Corresponding author: email address jdb@mail.nerc-essc.ac.uk

Abstract

GADS (Grid Access Data Service) uses the rapidly-growing and powerful Web Services framework to provide real-time access to large, gridded climatological datasets over the Internet. Data can be downloaded in a variety of popular formats (e.g. netCDF, HDF), irrespective of how or where the data are stored. The system provides richer functionality and greater flexibility than the existing DODS (Distributed Oceanographic Data System) standard, however, a DODS wrapper is provided for the benefit of users who already use this method. GADS implements two functions: one to enable users to enquire about what datasets are available and details about what they contain (dataQuery) and one to handle requests for data (dataRequest). There have been many recent developments to GADS, including improved metadata management and increased portability and extensibility. Future plans for GADS include migration to the Grid Service framework and the incorporation of GADS into workflows. Although GADS was developed with the oceanographic community in mind, it can be used as a method of accessing any gridded data sets.

1 Introduction

The use of increasingly high resolution computer models and observations of ocean and atmospheric circulation is driving a need for environmental scientists to be able to access large amounts of data rapidly and in a convenient and flexible manner. Many institutions generate and hold large sets of such data, but there are currently few standard and sufficiently powerful methods of accessing them. A common problem is that each institution tends to store data in a different form; there are many different file formats used in the climatological community (e.g. netCDF, HDF, GRIB and many non-standard formats) and data are discretized on a variety of grids.

Many projects include tasks to intercompare these large datasets (e.g. MERSEA, http://www.nersc.no/~mersea/, which is focussed on real-time ocean forecasting within Europe). An increasingly common activity is to compare the results of a computer model with satellite observations or indeed another model. These intercomparison tasks in particular highlight many of the problems currently faced by scientists. To perform general intercomparisons, two (or more) sets of data must be extracted from different data stores (where they may be stored in different formats), then cast onto a common grid, which might involve interpolation and rotating one of the datasets. These data handling tasks are often time consuming and tedious, but are very well suited to a distributed computing approach. The lack of established standards for describing data also provides diffi-

culties as it is not always clear as to whether two different descriptions of a dataset are semantically equivalent (see the NERC DataGrid project, http://ndg.badc.rl.ac.uk/).

With this in mind, we have developed a software system to address these problems. GADS (Grid Access Data Service, Woolf et al. (2003)) is a lightweight application that is deployed as a Web Service to allow data providers to expose their data stores. The development of GADS addresses the following major requirements:

- A need to abstract data from storage: With GADS, users do not need to know any details about how data are stored.
- A need to use standard names for variables, axes etc.: GADS provides a translation layer so that datasets can be exposed using standard names, irrespective of the internal representation.
- A need for platform independence: Through the Web Services framework, GADS can be invoked from a huge variety of programming languages and architectures.
- A need for compatibility with current advances in distributed computing: It is generally considered that Web Services (and their derivatives, Grid Services) will form the framework of distributed computing in the future. GADS will fit neatly into this framework (section 4.1).

The Distributed Oceanographic Data System (DODS, also known as OPeNDAP, University Corporation for Atmospheric Research (2003)) is currently commonly used to access oceanographic data by mapping URLs onto data files. Although operations such as file aggregation and data subsetting are possible, DODS suffers from many limitations, including only limited abstraction of data from storage and non-standard metadata queries. GADS represents (in many ways) an improvement to the DODS standard whilst retaining backwards compatibility with this existing system (see section 2.1).

2 The GADS System

2.1 GADS design principles

The most important aspect of GADS' large-scale architecture is that it is deployed as a Web Service. Web Services are a powerful framework for implementing distributed computing and have the major advantage that message passing to and from Web Services is done in a platform- and programming language-independent fashion. Furthermore, the description (WSDL, http://www.w3.org/TR/wsdl) and discovery (UDDI, http://www.uddi.org) of Web Services are standardised, and mechanisms for composing webservices into workflows (e.g. BPEL (IBM, 2003), myGrid: http://www.mygrid.org.uk/myGrid/web/components/Workflow) and ensuring security are emerging, thanks to a growing community of academic and business users and developers.

The GADS Web Service exposes two methods. One method (dataQuery) allows querying of the data holdings; that is, the datasets being served, which variables they contain, the geographical extent of the data, the details of the grids, and so forth. The second method (dataRequest) allows the user to make requests for and download actual data. These methods are described in detail in Woolf et al. (2003), but the important points to note are:

- Users prepare queries and requests for data without any knowledge of how the data are stored internally.
- Data are exposed with standard, CF (Climate and Forecast)-compliant variable names, irrespective of the naming conventions inside the data files.
- Data can be delivered in a many formats (netCDF, HDF, raw binary), irrespective of the form of the data files.
- The system behaves like an aggregation server. That is to say, if a user requests a set of data that spans several data files on disk, the user will retrieve a single file containing all the data.

GADS is a lightweight piece of software that can easily be copied and installed close (in network terms) to local data stores. In this way large amounts of data do not have to be transported wholesale to centralized data warehouses.

Via a translation layer (Woolf et al., 2003) data served by GADS can also be accessed through DODS URL-based queries, allowing backwards compatibility with current systems.

2.2 Web Portal

As part of the GODIVA project (Grid for Ocean Diagnostics, Interactive Visualisation and Analysis, Haines et al. (2003)), a web portal to GADS has been developed (figure 1). This interface was based on the appearance of the Live Access Server (LAS, Pacific Marine Environmental Laboratory (2003)) and allows users to graphically select datasets for download or visualisation. The portal uses GADS to extract data, which can then be rendered into still images or movies; this is done using other Web Services behind the scenes (figure 5).

3 Recent enhancements to GADS

Woolf et al. (2003) presented a prototype GADS system. There have been many recent improvements to this prototype:

3.1 Metadata storage and management

Appreciating that the management of metadata (i.e. the structured, detailed description of the data holdings) in a GADS system was of paramount importance, we have made considerable efforts to improve the efficiency and flexibility of metadata storage and management in the system. This has been achieved by implementing all the metadata-related functions in an internally-visible Web Service (figure 2).

By employing this architecture, the functions for querying, retrieving and managing metadata are loosely coupled from the rest of the system. This allows administrators to store metadata in the form of their choice. In the GADS prototype (Woolf et al., 2003), the metadata were stored as an XML file. This gives a simple, searchable and platform-independent means of storing metadata, but becomes less practical for large data holdings. If an administrator in charge of a large, GADS-enabled data store wished to use a high-performance relational database to store metadata, he or she would implement a set of functions (according to a specification) to interact with the metadata and expose these functions as a Web Service (which is only visible to the system, not the outside world). The rest of the system would use this Web Service transparently with no knowledge of the details of the metadata store. (We would anticipate making such Web Services for common systems (e.g. mySQL, XML) available, but this architecture allows users with more unusual requirements to be satisfied.)

A metadata management tool has been written to allow GADS administrators to perform routine tasks such as adding new datasets to the system without having to edit the metadata source directly. This interacts with the metadata via the internal metadata Web Service (figure 2), and therefore requires no knowledge of the metadata storage mechanism. It is written in Java to give platform independence; a screenshot is shown in figure 3.

3.2 Other improvements

The code has been rewritten in an object-oriented form to allow greater flexibility and extensibility. This new code is also more portable than the original prototype and has been tested successfully on Solaris and Linux systems.

Although other metadata representations are possible (section 3.1) it is anticipated that many GADS implementations will use an XML file to store metadata. With this in mind we have refined the XML metadata structure presented in Woolf et al. (2003). The new structure yields smaller XML files than previously which will improve search times. A UML representation of this new structure and an associated example XML snippet are shown in figure 4. As noted in the caption to this figure, this is only one possible structure for the metadata.

4 Future developments

There are many ways in which the development of GADS could be taken forward. We have identified the following as particularly fruitful areas of improvement:

- Currently the dataQuery method is limited in the nature of queries that it can handle. For example, it is not currently possible to construct complex queries such

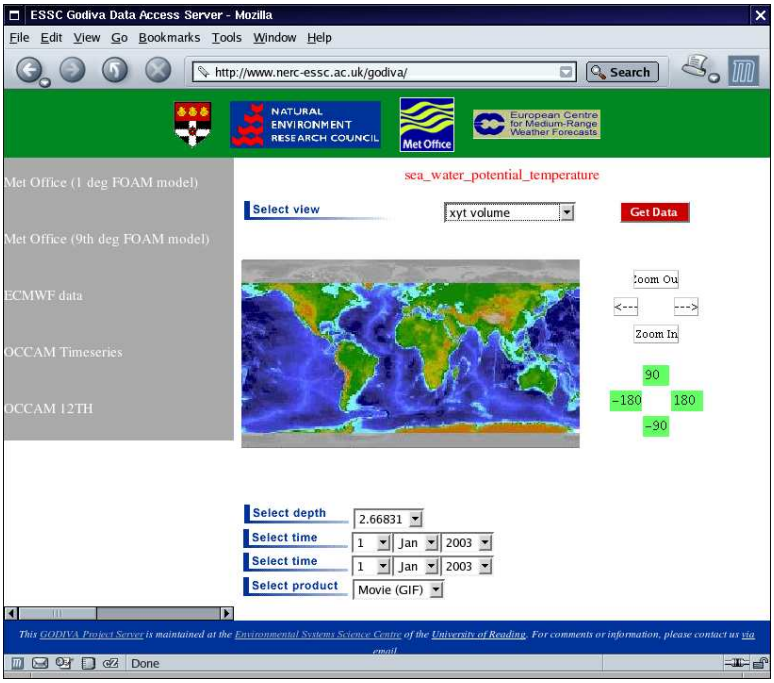


Figure 1: The GODIVA web portal for data access and visualization (http://www.nerc-essc.ac.uk/godiva). Through this portal, users can select data for download or for dynamic creation of still images or movies. Currently the portal provides access to data from the Met Office FOAM model, ECMWF and Southampton Oceanography Centre's OCCAM model. Certain datasets are protected by password access.

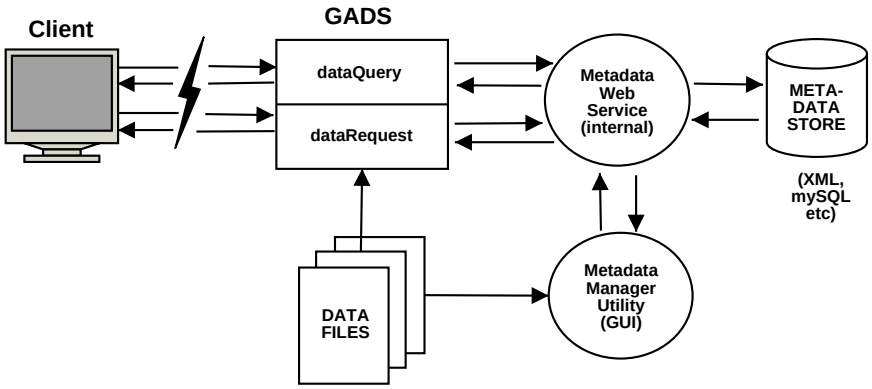


Figure 2: The large-scale architecture of GADS. Users send requests for metadata (dataQuery) and actual data (dataRequest) via a Web Service interface. Internally, GADS contains a metadata Web Service that handles all requests for metadata from both dataQuery and dataRequest (the latter needs to query the metadata to find the locations of data files). In this way, functions concerning metadata are loosely coupled from the system and therefore the metadata storage mechanism can be easily changed. The metadata manager utility (figure 3) is used to update the metadata, again via the metadata Web Service.

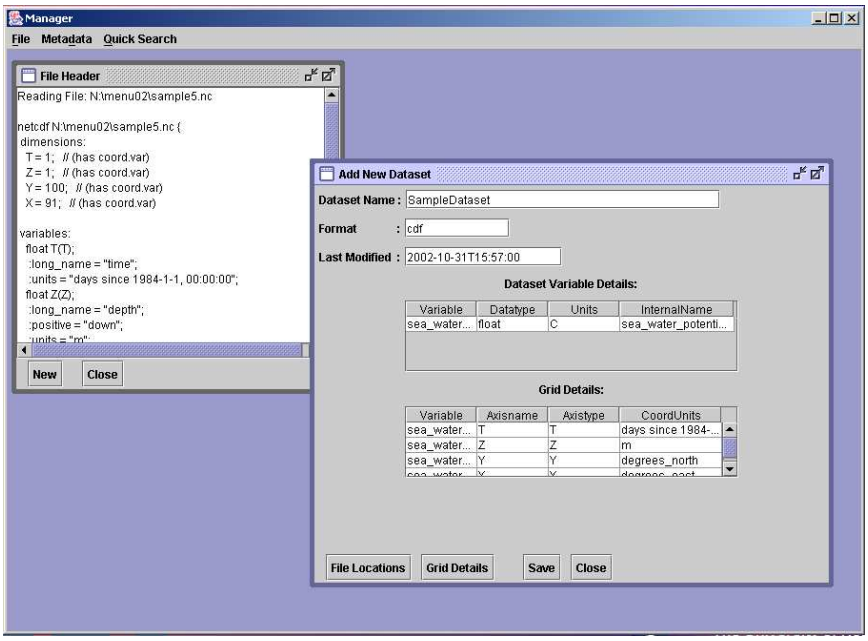


Figure 3: Screenshot of the metadata manager, showing the addition of a new dataset to the system. The utility is automated to a large extent; it can query the actual data files (figure 2) in order to discover the number of variables in the dataset, the grids on which the data exist, etc.

as "Please find all datasets containing salinity measurements in the South Atlantic". A move from RPC-style to document-style SOAP would allow richer queries to be processed.

- More output formats for dataRequest, in addition to the existing netCDF, HDF and raw binary formats need to be supported. Currently we are they working on adding support for GRIB output, and also for outputting data in VTK format for easy visualization.

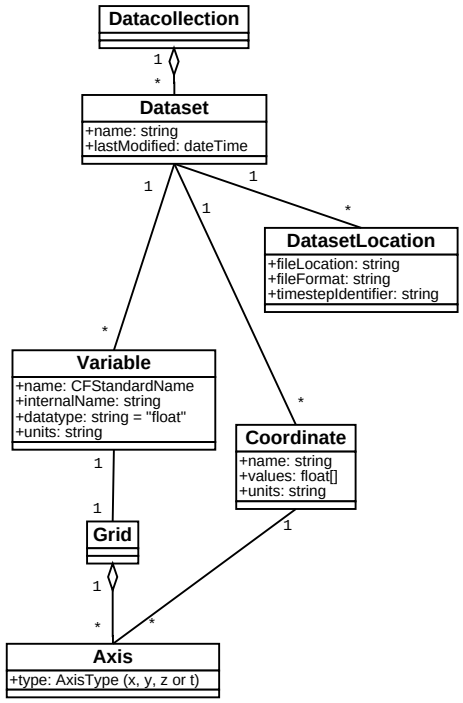
- The overall architecture should include support for plug-ins, so that support for new data formats can be added with minimal effort.

- There is a long-term migration path towards deploying GADS as a Grid Service (http://www.globus.org/ogsa). This would allow GADS instances to maintain internal state and to provide asynchronous notification to clients. For long extraction jobs, this would be very valuable; users could make queries about the progress of the extraction, the loading of the machine performing the job and receive automatic notification of completion.

4.1 Workflow

Whether as a Web or Grid Service, GADS will surely be used in conjunction with other Services in order to achieve certain tasks. For example, a typical workflow might involve extracting a set of data using GADS, passing the data to another service which performs some processing (for example, projecting ocean potential temperature measurements onto a density surface), and finally passing the processed data to a visualization service. The organization of a set of Web/Grid services into a pipeline or workflow is known variously as orchestration or choreography.

There are several challenges to be overcome in order to achieve this goal. Firstly, flexible and easy-to-use tools for assembling workflows need to be created. Specifications for different types of workflow engines are emerging (e.g. BPEL4WS, IBM (2003), SCUFL, The Taverna Project (2003)) but currently actual tools are sparse. We are working with members of the myGrid (http://mygrid.man.ac.uk) project



(a) UML diagram of metadata structure

```
<dataset name="FOAM_ONE_DEGREE" last_modified="2002-10-05T00:10:00">

    <variable name="sea_water_potential_temperature" datatype="float"
                                        units="C" internal_name="TMP">
      <grid>
        <axis type="t" coordinate="time"/>
        <axis type="z" coordinate="depth"/>
        <axis type="y" coordinate="lat"/>
        <axis type="x" coordinate="lon"/>
      </grid>
    </variable>

    <variable name="ocean_mixed_layer_thickness" datatype="float"
                                        units="m" internal_name="M">
      <grid>
        <axis type="t" coordinate="time"/>
        <axis type="y" coordinate="lat"/>
        <axis type="x" coordinate="lon"/>
      </grid>
    </variable>

    <datasetlocation>
        <file location="/data/netcdf/FOAM_02092001.0.nc"
                                        format="CDF" time="6454"/>
        <file location="/data/netcdf/FOAM_03092001.0.nc"
                                        format="CDF" time="6455"/>
        <file location="/data/netcdf/FOAM_04092001.0.nc"
                                        format="CDF" time="6456"/>
        ...
    </datasetlocation>

    <coordinate name="time" units="days since 1984-1-1">
       <value>6454</value>
       <value>6455</value>
       <value>6456</value>
       ...
    </coordinate>

    <coordinate name="lon" units="degrees_east">
       <value_array>
         <start>0</start>
         <step>1</step>
         <size>360</size>
       </value_array>
    </coordinate>

    <!-- also coordinates "lat" and "depth" -->

</dataset>
```

(b) Snippet of XML showing some metadata in this structure

Figure 4: The structure of metadata in GADS. Some relationships are particularly worthy of note. Each file location (in the DatasetLocation class) represents data from exactly one timestep, identified by the timestepIdentifier. Each variable exists on a Grid, which is made up of Axes; each Axis corresponds to one of the Dataset's Coordinates. Different variables in the same Dataset might exist on different Grids (it is common for temperature and velocity grids to be offset, for example) and have a different number of dimensions (as shown in the XML snippet). Regularly-spaced coordinates (typically latitude and longitude) are described using start-stride-count semantics, whereas irregularly-spaced coordinates (typically depth) have each possible value explicitly listed in the metadata (see the XML snippet). Note also the inclusion of the attributes "name" and "internalName" of the Variable class. This allows translation between standard, CF-compliant names and the names used to identify the variable in the data files: the XML snippet shows the internal name "TMP" exposed as the standard name "sea_water_potential_temperature". (N.B. It is important to note that this is not the only possible metadata structure. Since the metadata functions are loosely coupled from the rest of the system (figure 2) any metadata structure is possible as long as an appropriate internal Web Service can be created to query and update it. Other representations may be more efficient for certain applications, for example the storage of timeseries data.)

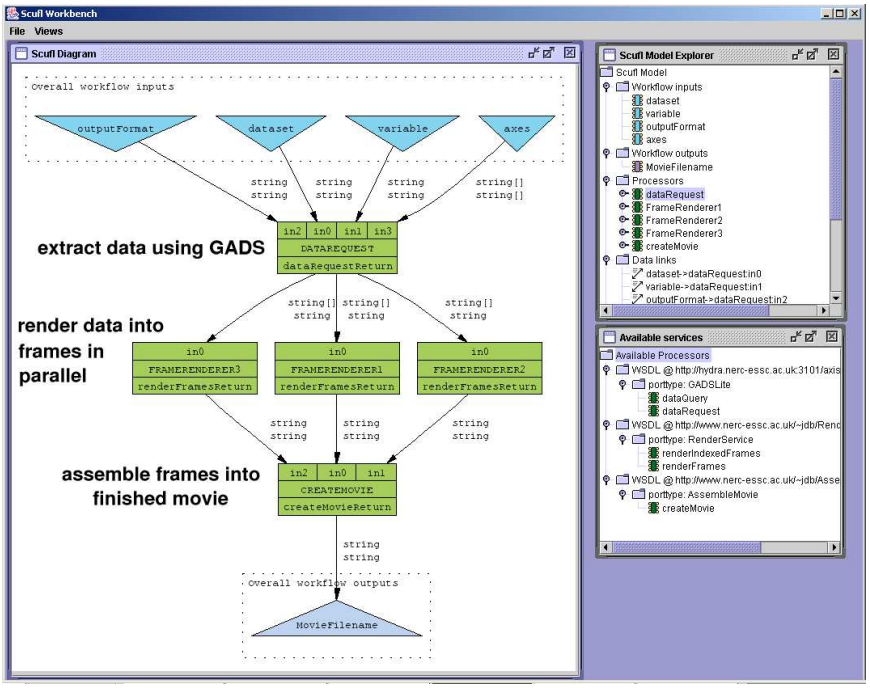


Figure 5: Screenshot of Scufl Workbench (http://taverna.sourceforge.net) showing GADS as the data extraction step in a visualization pipeline. The workflow is executed from top to bottom. This illustrative workflow has three parallel frame-rendering Web Services; these are sharing the load of turning the raw data extracted from GADS into movie frames. These frames are assembled into a movie using another Web Service. This illustration is very close to the workflow used behind the scenes to assemble dynamically-generated movies in the GODIVA web portal (section 2.2).

who have made significant advances in generating user-friendly, powerful tools for orchestrating Web Services (figure 5).

Another major challenge is the problem of how best to transport large amounts of data between Web/Grid Services. The gridFTP protocol provides a suitable transport mechanism but much thought needs to be devoted to the overall architecture of a Web/Grid Service-based distributed computing system in order to reduce unnecessary network traffic and overhead in a workflow. Imagine a situation where a scientist is assembling a workflow on a client machine and the workflow involves many remote Services, each of which consumes and produces large amounts of data. The data should be passed directly from Service to Service in a peer-to-peer manner, without having to pass through the client machine.

One way to achieve this is for each Service to output *pointers* to prepared datasets. It is these pointers which would be handled at the client side by the workflow engine. Each Service would then use the pointers to download datasets from some remote store before performing its task. GADS follows this model; upon requesting data using the dataRequest method, users do not receive the data directly, rather GADS outputs a URL to the extracted data set which can be downloaded using HTTP or gridFTP.

5 Conclusions

Although development is still very much in progress, GADS is already proving to be a powerful and flexible means of accessing large climatological datasets. As a Web Service, GADS can be accessed by a very large community of users, either via the GODIVA web portal (section 2.2) or by calling its methods directly through SOAP messaging. GADS can also be incorporated into workflows in order to achieve more complex data processing or visualization tasks (Sastry and Craig, 2003).

We hope that providers of climatological and oceanographic data will adopt GADS as a means of exposing their data and we are very happy to provide support for anyone who wishes to do so. By running GADS on their systems, institutions can serve their data to a wide community without having to distribute large amounts of data in a wholesale fashion to interested parties who, via GADS, can download exactly the data they want on demand.

Acknowledgements

This work was supported as part of the NERC GODIVA eScience project and also in part by the NERC Data Assimilation Research Centre (DARC, http://darc.nerc.ac.uk/).

References

Haines, K., J. Brooke, P. Challenor, A. Goddard, P. Killworth, and L. Sastry: 2003, Grid for Ocean Diagnostics, Interactive Visualisation and Analysis (GODIVA). *UK eScience All Hands Meeting, Nottingham University*.

IBM: 2003, Business Process Execution Language for Web Services Version 1.1 Specification. Online, http://www-106.ibm.com/developerworks/library/ws-bpel/.

Pacific Marine Environmental Laboratory: 2003, Live Access Server. Online, http://ferret.pmel.noaa.gov/Ferret/LAS/ferret_LAS.html.

Sastry, L. and M. Craig: 2003, Scalable application visualisation services toolkit for problem solving environments. *UK eScience All Hands Meeting, Nottingham University*.

The Taverna Project: 2003, Simple Conceptual Unified Flow Language (SCUFL) Version 0.1.beta4. Online, http://taverna.sourceforge.net/scuflfeatures.html.

University Corporation for Atmospheric Research: 2003, Distributed Oceanographic Data System. Online, http://www.unidata.ucar.edu/packages/dods.

Woolf, A., K. Haines, and C. Liu: 2003, A Web Service model for climate data access on the Grid. *International Journal of High Performance Computing Applications*, **17**, 281–295.