Available online at www.sciencedirect.com

ScienceDirect

Environmental Modelling & Software 23 (2008) 404–411

Environmental Modelling & Software

www.elsevier.com/locate/envsoft

A first approach to web services for the National Water Information System

Jonathan L. Goodall [a,*], Jeffery S. Horsburgh [b], Timothy L. Whiteaker [c], David R. Maidment [c,d], Ilya Zaslavsky [e]

[a] *Department of Civil and Environmental Engineering, University of South Carolina, Columbia, SC, USA*
[b] *Utah Water Research Laboratory, Utah State University, Logan, UT, USA*
[c] *Center for Research in Water Resources, University of Texas, Austin, TX, USA*
[d] *Department of Civil Engineering, University of Texas, Austin, TX, USA*
[e] *Spatial Information Systems Lab, San Diego Supercomputer Center, La Jolla, CA, USA*

Received 15 September 2006; received in revised form 15 January 2007; accepted 25 January 2007
Available online 25 October 2007

Abstract

A wealth of freely available hydrologic data are provided by governmental organizations including in situ observations, geospatial data sets, remote sensing products, and simulation model output. Despite having access to this information, much of the data remain underutilized in the hydrologic sciences due in part to the time required to access, obtain, and integrate data from different sources. Web services offer a means for sharing hydrologic data more openly by providing a standard protocol for machine-to-machine communication. We have used this new technology to create a machine accessible interface for the National Water Information System (NWIS), an online repository of historical and real-time streamflow, water-quality, and ground water level observations maintained by the United States Geological Survey (USGS). These services provide a middle-layer of abstraction between the NWIS database and hydrologic analysis systems, allowing such analysis systems to proxy the NWIS server for on-demand data access. We intentionally designed the services to be generic and applicable to other hydrologic databases, in order to provide interoperability between disparate data sources. Performance tests showed that, for time series with less than 1000 observations, the web services layer added minimal overhead in terms of data response time, and development of an example client application for time series visualization highlighted some of the benefits and costs of using web services for data access.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Hydrology; Water resources; Interoperability; Web services; Service-oriented architecture; Cyberinfrastructure

1. Introduction and problem statement

Hydrologists and water resource managers are able to access a large and increasing quantity of data for analysis, visualization, and modeling of the water environment. The number and variety of available data sources, however, make it difficult to quickly locate the most appropriate resource for a particular study. Furthermore, once the most appropriate data source has been identified, a considerable amount of effort is still required to reformat the data for use in analysis, visualization, or modeling software. As a result, researchers and practitioners spend a significant amount of time on basic data gathering and transformations, instead of scientific analysis and decision making.

This problem is most evident when one requires data collected by multiple individuals or agencies for a particular analysis task. The hydrology community, for example, requires meteorological forcing data from the weather and climate communities to drive watershed and ground water models. Atmospheric science communities, however, have different data models and data formats than those commonly used in the

* Corresponding author. Nicholas School of the Environmental and Earth Science, Duke University, Box 90328, Durham, NC 22701, USA. Tel.: +1 803 777 8184; fax: +1 803 777 0670.
E-mail address: jon.goodall@duke.edu (J.L. Goodall).

1364-8152/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.envsoft.2007.01.005

hydrologic sciences. If a hydrologist wants to make use of weather data in an analysis, it often first requires that he or she learn the file format or visualization tool used by that community. Therefore, interoperability of data between scientific subdisciplines, although necessary in terms of application (i.e. rainfall leads to runoff), remains cumbersome to implement.

While it is difficult to quantify the exact cost that results from a lack of interoperability, we believe it would be significantly reduced if hydrologic data were communicated between systems using a standard, machine accessible protocol instead of heterogeneous web pages (as is typically done now). In the science community, the challenge of interoperability is being addressed through the recent attention in cyberinfrastructure. A blue ribbon panel of the NSF suggested that the agency spend $1B to develop cyberinfrastructure in the sciences, stating that a comprehensive cyberinfrastructure to support scientific research could profoundly change the science and engineering communities (Atkins et al., 2003). Cyberinfrastructure plays a significant role in many geoscience sub-disciplines including ecology, atmospheric science, and hydrology (Bandaragoda et al., 2006; Green et al., 2005; Plale et al., 2006).

Although this work is towards the eventual goal of cyberinfrastructure for the hydrologic sciences, the particular focus of this paper is on the communication of hydrologic time series using web services. Web services provide the ability to pass messages between computers over the Internet, therefore allowing geographically distributed computers to more easily share data and computing resources. This is accomplished through a set of standard protocols that facilitate how a server documents its available services, how a client requests the server to perform a process, and how services are discovered on the web (Curbera et al., 2002). The standardization of web service protocols is impacting a wide variety of fields from business to science (Atkins et al., 2003; Foster, 2005; Hey and Trefethen, 2005; Stein, 2002).

Web services utilize extensible Markup Language (XML) as a common language for communicating between systems. While XML schemas have been proposed for basic communication using web services (e.g. SOAP and WSDL as described in Section 2.2), individual communities must supplement these generic protocols for particular domains. XML schemas have also been proposed for describing data in many geoscience communities (Atkins et al., 2003; Cox et al., 2002; Piasecki and Bermudez, 2003; Ramachandran et al., 2004); however, less work has been done on standardizing the language used for requesting hydrologic data through web services. Therefore, our focus is primarily on the language for requesting time series, instead of the language for describing a time series itself.

Web services have received increased attention in the geoscience literature. They are being used as the foundation for next generation environmental models (Mineter et al., 2003), for communicating data within sensor networks (Liang et al., 2005), and for building "virtual databases" where the data are distributed across multiple machines (Frehner and Brandli, 2006). Web services have been particularly popular in bioinformatics as a tool for sharing genomic data (Pillai et al., 2005; Stein, 2002; Sugawara and Miyazaki, 2003). Despite this pervious work in geosciences and bioinformatics, examples of web services in hydrology have been limited to date.

To investigate the application of web services for hydrology, we have prototyped a web service library for the National Water Information System (NWIS). NWIS is a United States Geological Survey (USGS) data repository containing over 250 million real-time and historical stream discharge, water-quality, and ground water level observations from the past 100 years (USGS, 2002) and is available over the Internet (http://waterdata.usgs.gov/nwis). The NWIS Web Services library we describe here wraps the logic for programmatically accessing and parsing NWIS web pages, making the NWIS data repository machine accessible through a standard web service protocol. This allows water modelers and software designers to incorporate routines for accessing the NWIS data repository into their own applications and, as a result, reduce the data gathering and maintenance responsibilities for the end user.

While this case study is limited in scope to a single data source that contains one or more hydrologic observation time series (i.e. a single fixed location where one or more variables have been or continue to be measured), the more long term goal is to create a generic standard for sharing hydrologic time series data that can be implemented for any hydrologic observations' database. Thus, in Section 3 we present a conceptual blue print for developing hydrologic time series web services that is not specific for NWIS. We then implement this design for NWIS, test the performance of the web services in terms of additional response time introduced by the web services layer, and lastly present an example client application of the NWIS web services with a discussion of the costs and benefits associated with using the web services library for data access.

This work is part of an ongoing effort by the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) Hydrologic Information System (HIS) group. It is meant to document our first attempts at creating web services for exposing remote hydrologic time series databases. For current efforts, the reader should visit the CUAHSI HIS website: http://www.cuahsi.org/his.

2. Background

2.1. The National Water Information System (NWIS)

The National Water Information System (NWIS) is the primary database for stream discharge and ground water level observations in the United States. It contains just under 1.5 million sites, of which approximately 350,000 have water-quality field observations, 24,000 have historical daily averaged streamflow observations, 8000 are collecting real-time observations, and 800,000 have ground water level observations (USGS, 2006). As of 1992, the total number of observations measured at these sites was over 250 million and included 62.7 million chemical observations, 181 million daily streamflow records, 635,000 flood peak discharges records, and 7.1 million ground water level observations (USGS, 2002). Data in the system range from the late 19th century to real-time observations collected every day.

The NWIS database is publicly accessible over the Internet through NWISWeb (http://waterdata.usgs.gov/nwis) and is available for public and private use, free of charge. Under the current NWISWeb design, a user can access any of the water-quality and quantity data by filling out a series of web forms that isolate a portion of the database. The user can then download the ASCII text file resulting from the query for further analysis. While this is a straightforward process, it can be time consuming, especially for first-time users who must learn the structure of the website, as well as for users who must make multiple requests to extract all data relevant for a study.

While NWISWeb is not explicitly intended for machine-to-machine communication, its design does allow one to automate the process of querying hydrologic time series. The algorithm for extracting data from NWIS is to (1) create a Universal Resource Locater (URL) that acts as a query on NWIS database (i.e. it describes the site, variable, and the period of interest) and (2) parse the ASCII file returned by the URL to extract a hydrologic time series. Once the web page has been accessed and downloaded, it can be parsed as if it were a local text file.

Despite the ability to programmatically read NWIS using the described procedure, there are a number of disadvantages with using this approach to build robust software systems. Perhaps the most significant disadvantage is its dependence on NWISWeb's interface remaining unaltered. Any changes that the NWIS administrators make to the logic for constructing URLs or for parsing web page responses will break all client applications that depend on the data. Therefore, one of the primary benefits of creating a web service for programmatic access to NWIS is to provide an intermediate layer between client applications that consume NWISWeb data, and the NWISWeb server itself. As NWISWeb evolves, forthcoming changes to the interface could be communicated to the NWIS Web Service administrators so that the service could adapt to those changes and minimize interruptions for client applications.

The web services we have developed for the National Water Information System are designed to be a generic means for accessing hydrologic time series data. As more earth science databases become machine readable to remote users, standards will be critical for insuring interoperability between the information and analysis systems. If the structure of web services for hydrologic time series can be agreed upon and standardized, then it will be possible for any data provider to create and maintain web services for their own data that follow the standards. These data providers will assume the responsibility for maintaining their own web services, but, because the services are standardized across providers, it will minimize the effort required by the scientists and practitioners to integrate data from multiple sources.

2.2. Web services

Web services consist of a set of protocols that together provide a mechanism for machine-to-machine communication over the Internet. While there are a number of protocols for web services, one of the most popular and the one used in this research is the Simple Object Access Protocol (SOAP). The general process flow of SOAP web services (Fig. 1) begins with a client learning of a service's methods from a Web Service Definition Language (WSDL) file. Then, the client uses the WSDL file to encode a SOAP request to the server. The server processes the SOAP request and sends any output resulting from the process back to the client machine. The standards behind SOAP web services are endorsed by the World Wide Web Consortium (http://www.w3.org/TR/soap) as well as by many commercial companies (Microsoft, IBM, and Sun to name a few), making their applicability far reaching in the information technology community.

One of the primary benefits of SOAP web service protocols over similar technologies for distributed computing (e.g. DCOM and COBRA) is interoperability across operating systems and programming languages (Umar, 2004). A web service written in Java and running on a Linux machine, for example, can communicate with a .Net application running on a Windows machine. This interoperability is achieved because all communication between servers and clients is encoded in XML-based languages and sent over the Internet (Curbera et al., 2002; Newcomer, 2002; Umar, 2004). XML is an operating system and programming language independent form of communication.

An application developer interacts with web services as with any other software library within their programming environment. The WSDL file that accompanies a web service includes documentation for creating a programming class on the client's machine. Most often, the developer will use a toolkit to read the logic within the WSDL file and create a class with methods described by the WSDL file. The developer is then able to work with the web service as if it were a local object, despite the fact that the object's methods are executed on a remote server instead of the client machine. Because of the availability of toolkits for accessing web services, the details of the XML messages that are passed between the client and server machines are typically hidden from the developer.

As with any technology, it is important to consider limitations and challenges associated with web services. We see the challenges for the hydrologic community to embrace web services as a means for communicating hydrologic data as both technical and institutional. On the technical side, automated

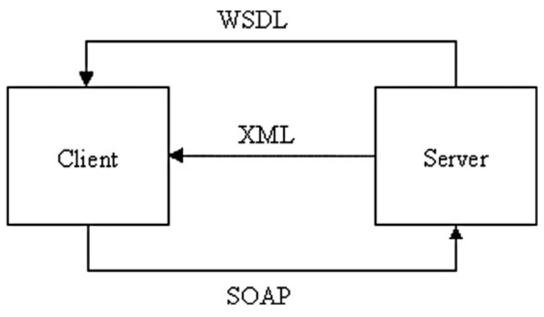


Fig. 1. Web services architecture.

access to hydrologic databases will likely result in more dependence on Internet connections for on-demand data access. Furthermore, hydrologic time series can become quite large, so a web services approach that relies on XML may require large bandwidths or techniques for compressing data within XML SOAP messages. A major institutional challenge that the community faces is the need to standardize web service signatures across data providers and consumers. Standardization is required to fully capture the potential for interoperability offered by web services.

The technical challenges are, for the most part, not unique to the hydrologic community and are being addressed by the larger information technology community as this technology continues to evolve. The political challenges (i.e. standardization), however, must be addressed by the hydrologic community because the particulars of how hydrologic information is organized and communicated can only be described by hydrologists themselves. Our hope is that, as the technology matures and as applications are developed that demonstrate the potential of web services and service-oriented architecture, the hydrologic community will embrace this technology and work towards establishing the standards necessary for interoperability.

3. Design concepts

We considered two primary questions in our design of web service for exposing a hydrologic time series database: (1) what methods are necessary to expose a database of hydrologic time series variables and (2) what arguments should be required of each method. We did not emphasize what each method should return to the user because this could be defined by existing XML schemas and, as discussed in Section 1, our focus is on the language for requesting time series. This design effort is a software engineering process of developing an Application Programming Interface (API) for retrieving time series from a database. The web service API must be generic enough to provide a consistent interface across heterogeneous sources, but it must also be specific enough to clearly isolate subsets of information from each source.

The basic conceptual framework for our design is based on the idea of a space–time–variable cube for referencing individual measurements according to where, when, and what was measured (Fig. 2). Maidment (2002) proposed this data cube concept as a means for integrating time series from heterogeneous data providers within a relational database, but it is also applicable to creating web services that are interoperable across data providers. By referencing data from multiple sources within this conceptual cube, it becomes possible to search for relevant data across providers; that is to say, who collected a sample is no more or less important than where the samples was taken, when it was measured, and what it is a measurement of.

In addition to the concept of referencing data in a space–time–variable cube, we believe it is also helpful for implementation purposes to consider two basic ways in which one might use a web service. The first is as a data discovery tool (e.g. "*Where has nitrogen been measured within the Neuse River?*"), and the second is as a data delivery tool (e.g. "*Get me the daily streamflow time series recorded for the Neuse River near Clayton, North Carolina site.*"). One reason for making this distinction between data discovery and data retrieval is that a user will likely interact with both types of queries very differently. Discovery services return a small amount of data, but are generally called more frequently than data delivery services because users are attempting to isolate a data set of interest. Delivery services, on the other hand, return larger data sets, but are called less frequently because users will have presumably already used the discovery services to isolate a time series of interest. In designing a system for exposing hydrologic time series, keeping these two types of services separate allows the system developers to maintain and implement the services in different ways. For example, to enhance performance of a mapping application, a discovery service may be implemented so that it accesses a local database of time series metadata, while a delivery service request would be routed to the original data source.

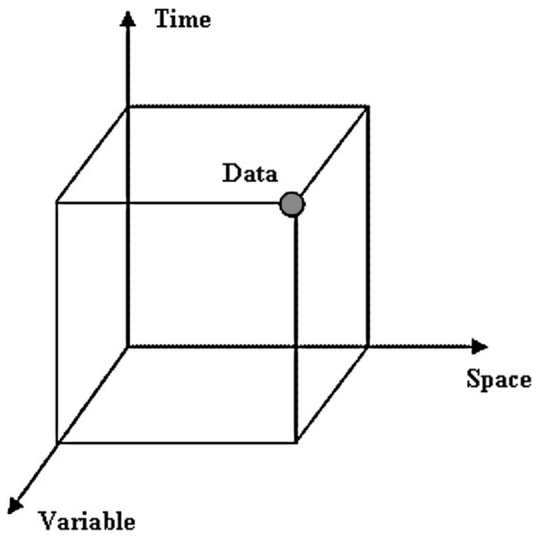


Fig. 2. The space–time–variable data cube.

This paper's primary focuses are on data delivery services because we see these as being the most critical for exposing the NWIS system via web services. The NWIS web services presented here were developed by wrapping the screen scraping process of building URLs and parsing the returning web page. In other words – we did not have inside access to the NWIS database. Because of this, we were limited in our ability to formulate custom search queries on the underlying database. Furthermore, based on the differences in how one might implement search and retrieval based services, a search algorithm might be implemented to read from local metadata files in order to improve performance. In this case, the data delivery services would be used to keep the local metadata files up-to-date, removing the need for searches on the remote database.

Using the concept of a space–time–variable cube and noting that our primary focus in this paper is data delivery services, we have identified three key data delivery methods necessary for exposing the NWIS database: GetSiteInfo, GetVariableInfo, and GetValues. The GetSiteInfo and GetVariableInfo methods allow a user to learn about the individual objects that comprise the space and variable axes of the cube (with site being a proxy for space). We did not include a method to learn about objects along the time axis (i.e. GetTimeInfo) as we do not see any need to provide metadata about each time step within the database. The third service, GetValues, essentially slices the cube for a particular site, variable, and time period of interest and returns the resulting time series. We only include this signature for data delivery because our model was the delivery of time series for a single location. Other signatures for the GetValues method, however, could be added in the future to return data for multiple sites at once (e.g. for a given variable, a given time, and all sites within a latitude/longitude box). Each method is described in more detail below.

The *GetSiteInfo* method returns a site object including basic metadata (Name, ID, Latitude, and Longitude) describing the site. The method requires a single input parameter that identifies the station according to the unique identification system used by the USGS for each of its approximately 1.5 million sites. Information returned from this method is in XML format with tags signifying the property names of each metadata attribute. This XML response can then be parsed or deserialized into an object for further analysis.

The *GetVariableInfo* method returns a variable object including basic metadata (Name, ID, and Units) describing a particular variable stored within the database. The method requires a single input parameter that uniquely identifies a variable according to the convention used by the USGS. Here we are not inventing a new variable typing system, but instead are unitizing the systems native to the data resource itself. The information returned by the method includes the name, units, and other basic metadata describing the variable being measured. The response is encoded in XML for transmission over the Internet.

Finally, the *GetValues* method returns a time series object that includes the values measured and the date and time of measurement. To request a time series object, the user specifies a site identifier, a variable identifier, a start date and time, and an end date and time. This time window is especially helpful for large time series with either a high temporal resolution or a long period of record. Again, the information is returned to the client application in an XML format (Fig. 3).

A prototype version of this service was built using Microsoft Visual Basic .Net and is hosted by the San Diego Supercomputing Center (SDSC) (http://river.sdsc.edu/NWISTS/NWIS.asmx). As this is an ongoing effort, the reader may wish to consult the CUAHSI Hydrologic Information System web page (http://www.cuahsi.org/his) for the latest version of the web services.

3.1. Key NWIS Web Service performance test

One concern of utilizing web services as an additional layer between client applications (analysis programs and hydrologic

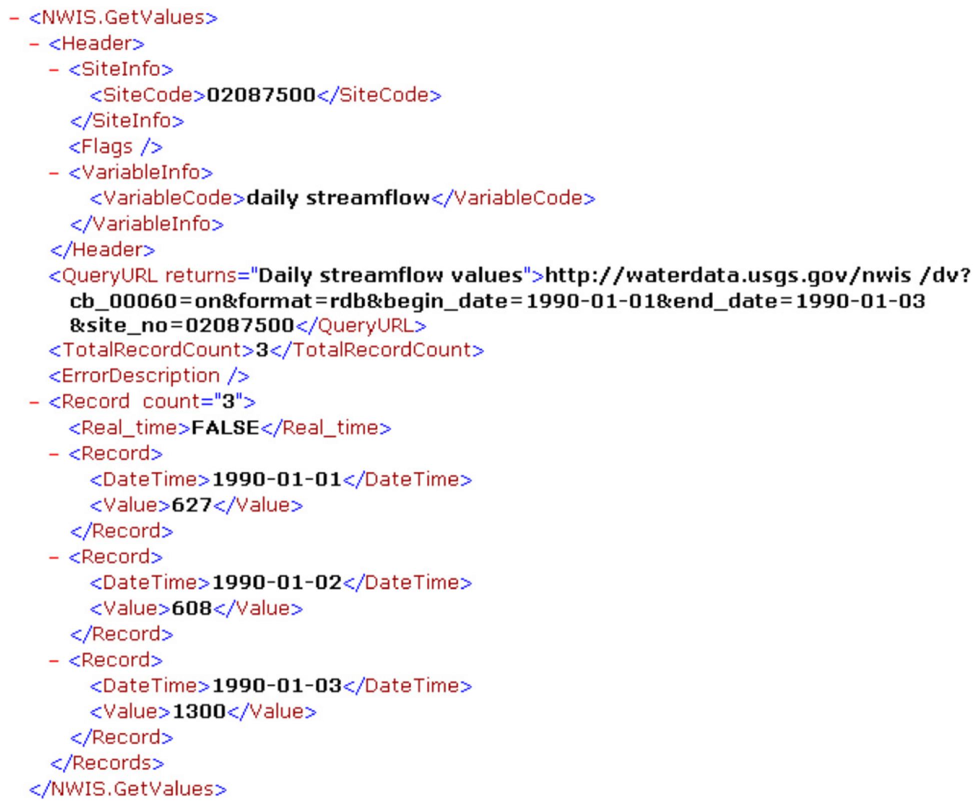

```
- <NWIS.GetValues>
  - <Header>
    - <SiteInfo>
        <SiteCode>02087500</SiteCode>
      </SiteInfo>
      <Flags />
    - <VariableInfo>
        <VariableCode>daily streamflow</VariableCode>
      </VariableInfo>
    </Header>
    <QueryURL returns="Daily streamflow values">http://waterdata.usgs.gov/nwis /dv?
      cb_00060=on&format=rdb&begin_date=1990-01-01&end_date=1990-01-03
      &site_no=02087500</QueryURL>
    <TotalRecordCount>3</TotalRecordCount>
    <ErrorDescription />
  - <Record count="3">
      <Real_time>FALSE</Real_time>
    - <Record>
        <DateTime>1990-01-01</DateTime>
        <Value>627</Value>
      </Record>
    - <Record>
        <DateTime>1990-01-02</DateTime>
        <Value>608</Value>
      </Record>
    - <Record>
        <DateTime>1990-01-03</DateTime>
        <Value>1300</Value>
      </Record>
    </Records>
  </NWIS.GetValues>
```

Fig. 3. Example time series XML document produced by the GetValues method for the USGS site Neuse River at Clayton, NC (02087500), daily streamflow for January 1–3, 1990.

models) and information systems is that the overhead, in terms of added response time, caused by the web service will negate any benefits derived from increased robustness and interoperability. Therefore, we conducted a test to quantify the additional access time introduced by the web service layer. To do this, we developed a simple client application in C#.Net that requested time series with increasing records lengths (1, 10, 100, 1000, 10,000, and the period of record of observations) from the USGS site Neuse River near Clayton, North Carolina (site code: 02087500). The client application requested data using two different methods: (1) our prototype NWIS Web Services, and (2) programmatically accessing and parsing NWISWeb pages. To minimize the impact of variations in server load on both the SDSC server hosting the web service and the NWIS web server in Reston, Virginia, each time series was requested 100 times using each method.

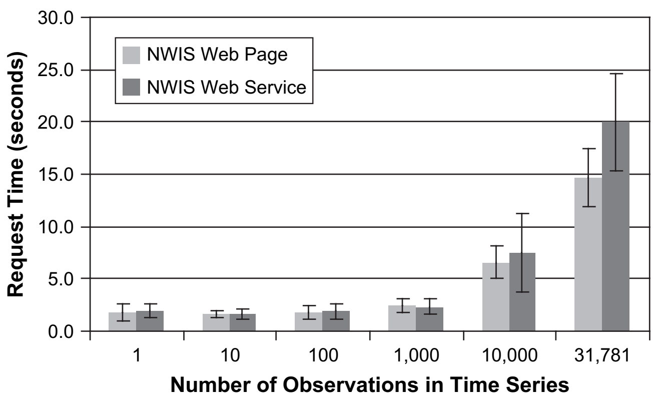
Our test showed that, for time series with records less than 1000 records, there is minimal overhead imposed by the web service approach (Fig. 4). The time required to return the time series using the NWIS Web Service was roughly the same as the time required for NWISWeb to create a web page and for the client to download and parse that page (between 1.5 and 2.0 s). For time series with over 1000 records, however, our test did show a performance cost associated with the web service approach for data access. For a time series of 10,000 records, our test showed that an NWIS Web Service request takes, on average, 7.5 s whereas requesting the data directly from NWISWeb takes 6.6 s. This performance cost increased to just over 5 s for a 32,172 observation time series (the period of daily streamflow record at the Neuse River near Clayton, North Carolina site) with the NWIS Web Service method taking 20.0 s to return the data and the NWISWeb method taking only 14.7 s.

Although it is difficult to isolate the exact reason for the performance cost associated with larger time series, part of the reason could likely be attributed to the increase in response memory sizes due to encoding objects in XML. An NWIS Web Service response, because it is encoded in XML, requires roughly twice as much memory as a web page returned directly from NWISWeb (Fig. 5). This suggests that for large time series, it may be useful for future designs to offer a compressed version of the time series values to keep performance similar to the baseline NWISWeb performance levels.

3.2. NWIS Web Service client example

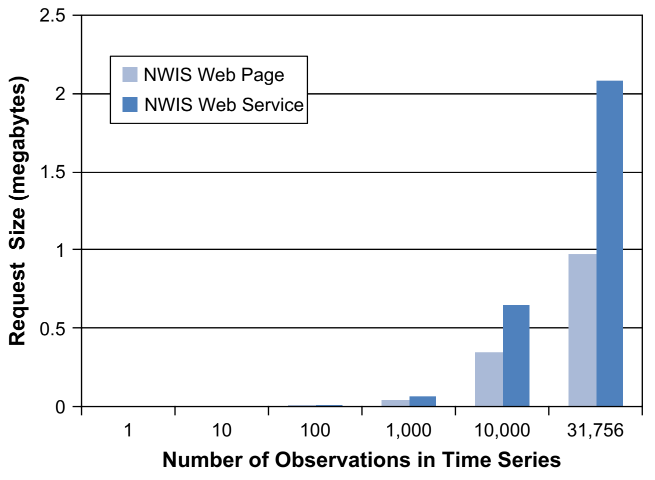
While the design of hydrologic time series web services is the focus of this paper, the power of web services are most obvious when incorporated into client applications. Applications that employ service-oriented architectures, where individual processes are accomplished by a set of web services distributed across multiple machines, are able to make use of generic, reusable components for tasks such as data access, data transformation, and data processing. Popular commercial software products used by hydrologists (e.g. Matlab® and Microsoft Excel) are able to utilize the web services, including those described here (Whiteaker et al., 2006). Having Matlab® input data using a web service instead of a local text file, for example, provides a means to directly couple data providers and consumers and, therefore, removes the intermediate data management responsibilities from the scientist.

To illustrate the benefit of web services from a user's prospective, we have modified an existing web-based application to be an NWIS Web Service client. The Time Series Analyst



Number of Observations in Time Series	NWIS Web Service Reponse Time (seconds)	NWIS Web Page Reponse Time (seconds)
1	1.9 ± 0.7	1.8 ± 0.8
10	1.6 ± 0.5	1.7 ± 0.3
100	1.9 ± 0.7	1.8 ± 0.7
1000	2.3 ± 0.7	2.4 ± 0.6
10000	7.5 ± 3.8	6.6 ± 1.5
31781	20.0 ± 4.6	14.7 ± 2.8

Fig. 4. Average time required to retrieve data using NWIS Web Services compared to NWIS web pages with error bars indicating ±1 standard deviation ($n = 100$).



Number of Observations in Time Series	NWIS Web Service Reponse Size (megabytes)	NWIS Web Page Reponse Size (megabytes)	Increase in Size Required for NWIS Web Service (%)
1	1.0E-03	3.0E-03	33%
10	2.0E-03	3.0E-03	67%
100	7.0E-03	6.0E-03	117%
1000	6.6E-02	3.7E-02	178%
10000	6.5E-01	3.5E-01	187%
31781	2.1E+00	9.7E-01	214%

Fig. 5. Memory requirements of NWIS Web Services responses compared to NWIS web page responses.

application was designed by researchers at Utah State University to provide Internet users the ability to plot and generate simple descriptive statistics for environmental time series data (streamflow, water-quality, climate, etc.) collected in the Bear River Watershed of northern Utah. In the original application (http://water.usu.edu/analyst/), the time series data are stored locally on a web server in a relational database. The user interacts with the Time Series Analyst via a web browser interface to request data. Upon request, the Time Series Analyst server side application queries the local database and uses the results of the query to generate plots, descriptive statistics, and data export files.

Subsequent to the creation and release of the NWIS Web Services, a new version of the Time Series Analyst was created that ingests data via the NWIS Web Services rather than from a local database (http://water.usu.edu/nwisanalyst/). The user interface of the new version is essentially the same, but rather than getting the data from a local database, the NWIS version of the Time Series Analyst sends requests directly to NWIS via the NWIS Web Services. This change means that data are requested by the Time Series Analyst and returned by the web services in real-time (Fig. 6). They not cached as an intermediate step within a local database (Fig. 6).

Both the local database and web services approaches have advantages and disadvantages. The primary advantage of the local database approach is speed. A user request for data is passed directly to the data repository (a local SQL Server database) and, because there is no need for network calls between the data repository and the application, performance is significantly better than the web services approach. Queries to a local database to extract a time series only take a fraction of a second whereas the NWIS Web Service calls, as discussed in Section 3.1, take at least 1.5 s and up to 20 s to return a time series.

On the other hand, the primary advantage of the web services approach is minimizing data management responsibilities and data redundancy for data consumers. All data archiving, data quality control and assurance, and data maintenance are done by the originating agency in the official data repository, and any changes to the official repository are immediately available to all users. Better access to hydrologic data could mean using the services “in real time” to visualize remote data sets, but it could also mean the services are used to keep local databases up-to-date as new data become available. Ultimately, how the web services are put into practice, for on-demand access or for automated data harvesting, will be determined by an organization’s needs and resources.

4. Conclusions

In this first attempt at creating web services for exposing the National Water Information System, our goal was to provide a standard, machine accessible interface to the federal data archive that followed industry standards for web services (i.e. SOAP, WSDL). We intentionally designed the interface to be generic so that it may be implemented as a standard protocol for exposing any hydrologic time series database. While it is difficult to quantify the benefit of web services for data access and interoperability, our assumption is that automated access to hydrologic data through standardized interfaces will reduce the time required by scientists to incorporate data into analysis applications.

Exactly how the web services are used by researchers is open. We envision that in some cases, the web services may

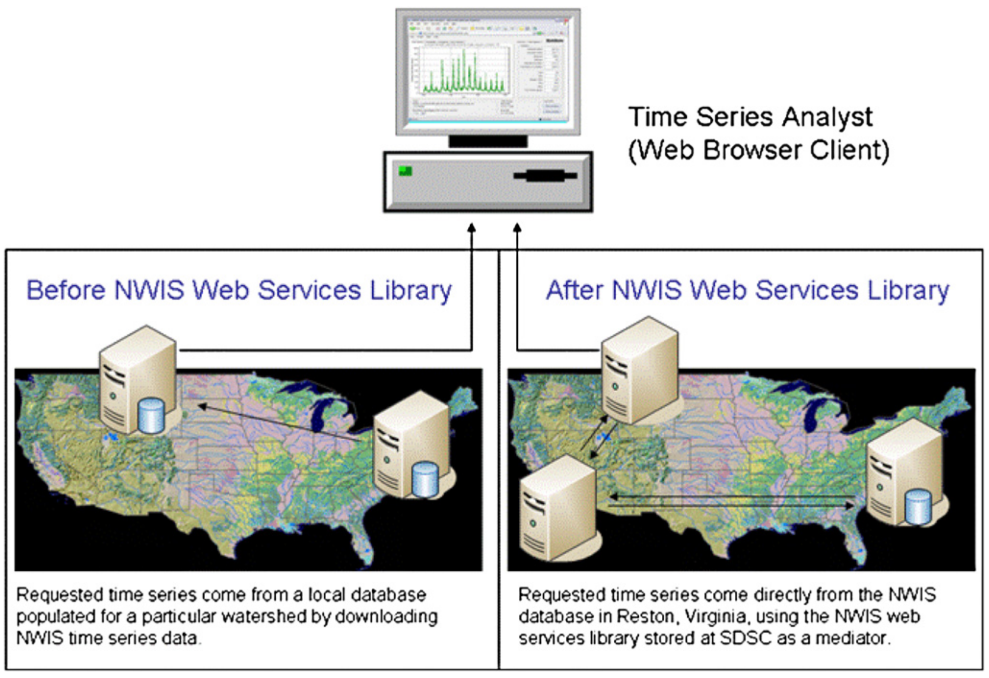

Time Series Analyst
(Web Browser Client)
Before NWIS Web Services Library
After NWIS Web Services Library
Requested time series come from a local database populated for a particular watershed by downloading NWIS time series data.
Requested time series come directly from the NWIS database in Reston, Virginia, using the NWIS web services library stored at SDSC as a mediator.

Fig. 6. Data flow for the Time Series Analyst before and after becoming an NWIS Web Service client.

be used to keep a local achieve up-to-date, while in other cases, the web services may be used for direct data access within an analysis or modeling routine. Because the NWIS Web Service presented here is built on industry standards like SOAP and WSDL, the services can be used by any software systems able to consume web services (Matlab®, Microsoft Excel, etc.). This makes web services an attractive option for distributed scientific computing.

Performance tests showed that time series from NWIS with less than 1000 observations take approximately 1.5–2 s to retrieve. It is important to note that this response time is a result of the NWISWeb response time – the service layer adds little overhead. For larger time series ($<$1000 observations), however, the service layer does add overhead. We hypothesize that the overhead is due to the verbosity of XML. Future work should test this hypothesis and, if true, proposed compression techniques to minimize the transmission size of large time series using web services.

This work is a first step towards creating cyberinfrastructure for the hydrologic sciences through CUAHSI Hydrologic Information System. Future efforts will be directed towards refining the design the web services presented here and implementing web services for other federal data repositories. Given the increased availability of digital data and the time required to access and integrate data from heterogeneous sources, we believe that this is a critical need for the hydrologic sciences. If a standard can be proposed and implemented for sharing hydrologic time series using web services, it will greatly increase the interoperability of hydrologic data and allow scientists to more easily utilize these data sets in scientific studies in the coming years.

Acknowledgement

This work was supported by the National Science Foundation (NSF) under grant #0413265.

References

Atkins, D.E., Droegemeier, K.K., Feldman, S.I., Garcia-Molina, H., Klein, M.L., Messerschmitt, D.G., Messina, P., Ostriker, J.P., Wright, M.H., 2003. Revolutionizing Science and Engineering Through Cyberinfrastructure: Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure. Available from: http://www.communitytechnology.org/nsf_ci_report/ (accessed 05.01.07.).

Bandaragoda, C., Tarboton, D., Maidment, D., 2006. Hydrology's efforts toward the Cyberfrontier. EOS, Transactions AGU 87 (1), 2.

Cox, S., Cuthbert, A., Lake, R., Martell, R., 2002. Geography Markup Language (GML) 2.0 OpenGIS Implementation Specification. Available from: http://portal.opengeospatial.org/files/?artifact_id=11339 (accessed 05.09.06.).

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S., 2002. Unraveling the Web services Web: aintroduction to SOAP, WSDL, and UDDI. IEEE Internet Computing 6 (2), 86–93.

Foster, I., 2005. Service-oriented science. Science 308 (5723), 814–817.

Frehner, M., Brandli, M., 2006. Virtual database: spatial analysis in a web-based data management system for distributed ecological data. Environmental Modelling & Software 21 (11), 1544–1554.

Green, J.L., Hastings, A., Arzberger, P., Ayala, F.J., Cottingham, K.L., Cuddington, K., Davis, F., Dunne, J.A., Fortin, M.J., Gerber, L., Neubert, M., 2005. Complexity in ecology and conservation: mathematical, statistical, and computational challenges. Bioscience 55 (6), 501–510.

Hey, T., Trefethen, A.E., 2005. Cyberinfrastructure for e-science. Science 308 (5723), 817–821.

Liang, S.H.L., Croitoru, A., Tao, C.V., 2005. A distributed geospatial infrastructure for Sensor Web. Computers & Geosciences 31 (2), 221–231.

Maidment, D.R., 2002. Arc Hydro: GIS for Water Resources. ESRI Press, Redlands, Calif, 203 pp.

Mineter, M.J., Jarvis, C.H., Dowers, S., 2003. From stand-alone programs towards grid-aware services and components: a case study in agricultural modelling with interpolated climate data. Environmental Modelling & Software 18 (4), 379–391.

Newcomer, E., 2002. Understanding Web Services: XML, WSDL, SOAP, and UDDI. Addison-Wesley, Boston, xxviii, 332 pp.

Piasecki, M., Bermudez, L., 2003. HYDROML: Conceptual Development of a Hydrologic Markup Language. IAHR Congress, Thessaloniki, Greece.

Pillai, S., Silventoinen, V., Kallio, K., Senger, M., Sobhany, S., Tate, J., Velankar, S., Golovin, A., Henrick, K., Rice, P., Stoehr, P., Lopez, R., 2005. SOAP-based services provided by the European Bioinformatics Institute. Nucleic Acids Research 33, W25–W28.

Plale, B., Gannon, D., Brotzge, J., Droegemeier, K., Kurose, J., McLaughlin, D., Wilhelmson, R., Graves, S., Ramamurthy, M., Clark, R.D., Yalda, S., Reed, D.A., Joseph, E., Chandrasekar, V., 2006. CASA and LEAD: adaptive cyberinfrastructure for real-time multiscale weather forecasting. Computer 39 (11), 56–64.

Ramachandran, R., Graves, S., Conover, H., Moe, K., 2004. Earth Science Markup Language (ESML): a solution for scientific data-application interoperability problem. Computers and Geosciences 30 (1), 117–124.

Stein, L., 2002. Creating a bioinformatics nation – a web-services model will allow biological data to be fully exploited. Nature 417 (6885), 119–120.

Sugawara, H., Miyazaki, S., 2003. Biological SOAP servers and web services provided by the public sequence data bank. Nucleic Acids Research 31 (13), 3836–3839.

Umar, A., 2004. The emerging role of the Web for enterprise applications and ASPs. Proceedings of the IEEE 92 (9), 1420–1438.

USGS, 2002. NWISWeb: New Site for the Nation's Water Data. Available from: http://pubs.usgs.gov/fs/fs-128-02/ (accessed 09.09.06.).

USGS, 2006. NWISWeb Home Page. Available from: http://nwis.waterdata.usgs.gov (accessed 15.09.06.).

Whiteaker, T.L., Beran, B., Goodall, J.L., Min, T., Tarboton, D., To, E., Valentine, D., 2006. CUAHSI HIS Web Services Workbook. Available from: http://www.cuahsi.org/his/docs/HIS-workbook-20061130.pdf (accessed 05.01.07.).