On-Line Data Archives

K.A. Hawick, P.D. Coddington, H.A. James and C.J. Patten

Distributed and High Performance Computing Group
Department of Computer Science, University of Adelaide
Adelaide, SA 5005, Australia
and
School of Informatics, University of Wales, Bangor
Dean Street, Bangor, LL57 1UT, UK &
Email: *hawick@bangor.ac.uk*
Tel: +44 1248 382717, Fax: +44 1248 361429

Abstract—

Digital libraries and other large archives of electronically retrievable and manipulable material are becoming widespread in both commercial and scientific arenas. Advances in networking technologies have led to a greater proliferation of wide-area distributed data warehousing with associated data management challenges. We review tools and technologies for supporting distributed on-line data archives and explain our key concept of "active" data archives, in which data can be processed on-demand prior to delivery. We are developing wide-area data warehousing software infrastructure for geographically distributed archives of large scientific data sets, such as satellite image data, that are stored hierarchically on disk arrays and tape silos and are accessed by a variety of scientific and decision support applications. Interoperability is a major issue for distributed data archives and requires standards for server interfaces and metadata. We review present activities and our contributions in developing such standards for different application areas.

Keywords: active data archive; data warehouse; distributed data; Java; CORBA; middleware; digital library.

I. Introduction

Digital libraries and other large archives of electronic data are becoming widespread in both the commercial and scientific arenas. This has led to a real practical need for software systems to manage large data archives. Such systems should support efficient and robust data ingest, storage, cataloging, management, browsing and/or querying, processing, and access. Advances in networking technologies and the enormous growth in use of the Internet and the World Wide Web have led to a greater emphasis on making data archives available on-line. This enables access to a wealth of data in multiple data archives distributed throughout the world, which is extremely useful for a for a variety of commercial and scientific applications, particularly in areas such as data mining and decision support.

The enormous potential benefits of wide-area distributed data warehousing have not yet been fully realised, mainly due to a lack of software tools for supporting wide-area distributed data warehousing and for developing applications to effectively use these on-line data archives. There are additional challenges associated with managing and accessing multiple data archives that are distributed over wide-area networks, in terms of both software support and data management policy. Performance, security, authentication, electronic commerce, portability, transparent access, and interface standards all become major issues.

Standard data warehouses provide services for users and client applications to query, select and download data. We have been working on mechanisms to support on-line data archives that can also provide on-demand processing of data before delivery to the client. These "active" data archives are particularly useful for scientific and decision support applications, and can greatly enhance the value of static collections of large datasets.

For example, in a decision support application, a single query may require substantial processing of large amounts of data, potentially from multiple distributed data archives, in order to produce an information product that may be very small (possibly just a single number or a yes/no answer) compared to the amount of data that was processed. In many cases it is infeasible or very inefficient to transfer all the necessary data across a wide-area network to be processed on the client. A better approach is to do some (or all) of the processing on the server side, particularly if it involves data reduction operations.

Active data archives enable data providers to sell value-added information as well as (or instead of) raw data. The addition of processing services to on-line data archives allows users to request value-added derived data products on demand. If the data processing is fairly simple, it may be possible to preprocess all the raw data and store the derived data products, however this will generally require too much computational and storage resources, particularly for large data sets such as satellite data archives. A better approach is to generate the derived data products only when they are requested, although once the processing has been done, it may be useful to cache the results for future use. This is feasible if a suitable data naming scheme can be devised.

Active data archives require additional computational infrastructure to support services for processing data on the server before delivery. The client must be able to easily specify the processing that is required. This is straightforward for a single process acting on a single data product, however more demanding applications may need to request a sequence of operations on multiple data products. The different data sets and software packages required for a given task may

be located on different machines, possibly distributed over a wide-area network. For applications that require interactive or near real-time results, the processing requirements may be so great that the data must be transferred to a high-end compute server. Complex tasks may require careful scheduling of storage and computational resources to achieve the fastest turn-around time for these requests.

In order to handle all these requirements, an active data archive needs a sophisticated software and hardware infrastructure to support distributed and high-performance computing applications. The software technology for building such an infrastructure has greatly improved in the last few years, with the development of the World Wide Web and the availability of platform-independent, distributed, object-oriented programming environments such as the Common Object Request Broker Architecture (CORBA) [25] and Java [36]. These technologies are examples of what is known as distributed computing middleware, which provides a standardised programming interface that abstracts over low-level details, allowing much easier development of distributed computing applications that can work across different computer and network architectures and systems software.

These are the main technologies we are using in our research work, which has focussed on developing a high-level software infrastructure to support distributed active data archives. Our software is designed to act as distributed computing middleware, but at a much higher level than Java and CORBA, which we have used as building blocks. Our high-level middleware infrastructure is actually an object-oriented metacomputing [14, 15, 34] environment, which aims to provide transparent and efficient access to services for querying, processing and accessing data on multiple computers distributed across a wide-area network.

Our work has focussed mainly on archives of large (bulk) data sets, such as scientific data and satellite imagery, however the main concepts are more generally applicable. Other applications areas include those using archives of financial data; meteorological data (measurements and simulation results); agricultural and environmental data; and other areas that involve large amounts of relatively slowly changing data intermixed with smaller derived data products.

Achieving near real-time performance for applications that use distributed archives of bulk data is a challenging problem. The data is often stored on tape silos with large access overheads, processing may be computationally intensive, and downloading large amounts of data over wide-area networks can be very slow. A lot of our research effort is targeting these problems, searching for ways to utilise high-speed networks, computers and storage systems where they can really make a difference; developing intelligent data caching and process scheduling strategies; and designing an efficient middleware infrastructure for supporting distributed active data archives.

A key criterion for success is that client applications should not require the end user to be an expert in the technologies that enable access to the data, but only that users are able to specify, or describe, the data and processing they require, using an appropriate, intuitive, text-based or graphical user iterface. The end user should not have to worry about details such as specifying where to find the data, identifying the server with the fastest access time if the data is stored in multiple data archives, finding the fastest available machine that has the right software for doing the processing, resubmitting the request if there is a fault in the client or the server or the network, etc. These are problems that should be handled by the middleware infrastructure.

A well designed active on-line archive system will enable various geographically distributed data custodians to pool their resources and provide an integrated interface to the data collections. This is particularly useful for providing derived data such as executive summaries; access and usage statistics; high level views of the collections and summary information of use to the custodians themselves as well as end users.

There are in summary four key components to an active data archive. The low level bulk data storage management system (for both data and metadata or index data); the middleware data transport and service brokering component; the processing services which may be high performance computing platforms; and the client that will act as a suitable interface to the whole collection. Our approach to this problem has been to integrate together as many standard components as possible, providing some key middleware (developed using both Java and CORBA) at the server side and an extensible client interface (using predominantly Java) that can run from a Web browser environment.

In this paper, we provide a summary of our research work on developing a middleware infrastructure for supporting distributed active data archives, for use in various scientific and decision support applications. In section II, we review the technologies that are available for wide-area distributed data warehousing, and what capabilities they provide for supporting active data archives. We also present examples of some prototype active data archives that we have created, to illustrate the capabilities and limitations of these technologies. Section III gives a brief overview of our DISCWorld project, which is developing a high-level middleware infrastructure to support distributed active data archives. Section IV discusses the key issue of distributed data storage, which underpins all our work on data warehousing and which is designed to interoperate with the DISCWorld middleware. Section V discusses some evolving standards for metadata and interfaces for querying, processing and delivery of data from on-line data archives. Finally, we summarise the main issues faced in building a middleware infrastructure for supporting distributed, active, on-line data archives, and the problem areas that are the subject of current research.

II. Enabling Technologies for Data Warehousing

There is a wide spectrum of technologies available with which on-line data warehouses may be constructed, managed and accessed. In this section we focus on the distributed computing technologies that enable access to data and processing services from on-line data archives, and provide examples of the use of these technologies in some prototype active data archives that we have developed.

A. World Wide Web

The development of the World Wide Web provided a simple mechanism for accessing data over wide-area networks using the hypertext transfer protocol (HTTP) to download files from Web servers. The use of hypertext enabled the development of user-friendly, point-and-click Web browsers, which have rapidly become a ubiquitous standard interface to on-line information. The Web has therefore become a standard platform for hosting and interfacing to digital libraries and on-line data archives.

Active data archives require a mechanism for invoking programs for processing data on the server side. HTTP provides this capability via the Common Gateway Interface (CGI) [17], which allows server-side programs to be called using an HTML form or a URL in a special format that encodes the program and the parameters being sent to it. The CGI program generates output data (usually a Web page in HTML format, but in principle this can be any kind of data) and returns it to the browser, where it is handled just as if it were from a normal file sitting on a Web server and accessed via a standard URL. HTTP and CGI thus provide a simple but powerful mechanism for implementing active data archives. Since most on-line data archives are accessed via Web servers, this type of interface has become very common, particularly for text-oriented digital libraries, but also for image-based applications such as serving maps on demand. The Open GIS Consortium [26] has been working on standards for Web-based mapping [28], and the initial proposals are based on HTTP and CGI.

One of the first on-line data archives that we created was a repository of satellite data [21] from the Japanese GMS5 geostationary meteorological satellite, which provides hourly images in 4 spectral bands for the entire hemisphere of the Earth visible from the satellite. We developed a simple satellite data browsing system known as ERIC [21, 23], which used Web technology to interface to the satellite archive. This prototype active data archive interface was developed partly to help some of our collaborators make better use of the GMS5 repository for scientific applications, and partly to explore the software issues in constructing wide-area on-line data archives with support for server-side processing.

The simple queries supported under our initial ERIC system were for a particular spectral band at a particular date and time. Since the full images are very large (around 2300x2300 pixels), we also allowed the user to request a specific image resolution and a particular region of interest. Later we added capabilities for additional server-side processing, such as producing time-series animations as MPEG videos. This takes a substantial amount of computation, but we were able to reduce the processing time by distributing the workload over a cluster of workstations. The ERIC system was originally implemented as a single driver program running on the Web server machine and invoked as a CGI program by the Web server daemon. The driver script was a Perl program, and some of the necessary functions were implemented as C programs or shell scripts invoked as system calls from the Perl script. The user interface was a simple static HTML form.

This system architecture allows for rapid prototyping of applications, but has a number of disadvantages. Debugging, error handling, monitoring of processes and maintaining state information are all more difficult when the client has to interface to the server program using CGI, which makes it hard to develop robust CGI applications. Perl is an excellent language for rapid prototyping, but not particularly good for developing and maintaining large and complex software projects. Embedded system calls are not very efficient either in memory or in computational startup costs, since they must employ a separately spawned UNIX process to handle them.

These problems drove us to develop the improved prototype systems described in the following sections. In particular, our DISCWorld middleware and storage systems described in sections III and IV were designed to address a number of the design and software engineering issues identified in implementing the ERIC server. The issue of placement of ERIC processes was particularly difficult. Distribution was achieved through statically configuring a remote processor to provide some of the computation necessary to fulfill a query. Ideally we would like a middleware system to provide a service to manage this more flexibly through a proper job scheduling component.

B. Java Clients

The ERIC client presented users with a familiar HTML forms interface. This type of interface is functional, but can be restrictive, since it cannot easily be made more user-friendly to allow easier navigation through the data archive, or customised for specific purposes. A much richer and more interactive user interface can be provided by using JavaScript or a Java applet. Interfaces of this kind are now commonly used for Web-based mapping applications that access geospatial data from Web servers, including applets based on the OpenMap Java toolkit [2] and a number of other examples from the OpenGIS Web Mapping Testbed [?]. Development of ERIC progressed from a simple forms-based CGI program to using a basic Java applet client that allowed some limited client-side interaction such as rubber-banding a region of interest. The next step was a Java version of ERIC that ran inside the DISCWorld client environment (screenshots of this version are shown in figures 1 and 2). The DISCWorld version of ERIC enabled much greater user interaction with the system, making it easier to specify server-side image processing operations, and allowing the user to access and reuse intermediate results in a processing chain.

We also developed a prototype satellite imagery viewer applet [6] that used the Java Advanced Imaging (JAI) API to provide support for image tiling, improved interactive navigation through a large tiled data set, and client-side image processing. The client could access data either from a Web server (using a CGI or a standard URL) or from an image database using standard CORBA interfaces (described in sections II-C and V). The use of JAI and CORBA made this a much more heavyweight client than the basic Java applets used for ERIC, which caused some stability problems due to the still relatively immature nature of this technology. This can be an area of concern for complex applets with a lot of client-side processing.

We also developed a Java applet to provide a powerful, efficient and intuitive graphical user interface to a large archive of images of the human body produced by the Visible Human

Project [41]. Our Visible Human Viewer [4, 9] was one of the first demonstrations that a Java applet could provide an excellent Web-based graphical user interface to a large on-line data archive, since they can be more interactive, responsive and easy to use than a simple HTML-based interface.

The Visible Human Viewer allows the user to download low, medium and high resolution images. Since downloading the high resolution images can take a long time, we have added the capability for the user to select only a specified region of the image by rubber-banding a thumbnail image. The applet then accesses a server program, using either CGI or Java Remote Method Invocation (RMI), which uncompresses the image, crops out the specified region in the image, compresses it back to JPEG format, and returns it to the client. This is a very simple example of how an active data archive can make accessing large data sets more efficient.

An advantage of using Java applets is that they can easily be developed to allow customised user interfaces to be downloaded for different users or different applications, based on a specified set of user preferences. This would be particularly useful for complex decision support applications.

C. Other Implementation Technologies

CORBA [25] and Java Remote Method Invocation (RMI) [36] offer more powerful and flexible interfaces to server-side processing than can be achieved using HTTP and CGI. They are both targeted at distributed computing applications, and are therefore excellent environments for developing distributed active data archives. They are particularly effective for complex applications requiring server-side or distributed data processing. They also offer in-built support for multiple distributed data archives, since they allow for registries that advertise available data access and processing services. This means clients do not need to know exactly what server is hosting a particular service, but can query a registry to find a required service.

Another advantage of using Java and CORBA is that they allow the development of both client and server programs using a modern object-oriented programming approach, which makes it easier to program larger, more complex software systems; to specify standard interfaces to services; and to handle data access or processing errors. Client applications that are written in object-oriented languages, e.g. Java applets, can interface to the data archive in a more natural and flexible object-oriented manner, rather than having to drop down to a lower level to manually convert client requests to a set of parameters that can be passed to the server CGI program using HTTP.

Many large distributed systems have been designed using CORBA's Interface Definition Language (IDL) [25] to specify the interfaces between different parts of the system, in which case it is usually easier to use CORBA to handle distributed processing. For pure Java applications, remote processing services are more easily implemented using Java RMI rather than CORBA. RMI is particularly useful for applications with Java applet client interfaces. CORBA is more suitable for developing distributed versions of legacy applications, since it was designed to interface between programs written in different languages. Java can handle programs written in other languages by using the Java Native Interface (JNI), however this process can be non-trivial. Developing distributed systems using both Java and CORBA allows the programmer to use the strengths of each system in particular areas. Future versions of these technologies (Java 2 Enterprise Edition and CORBA 3) should make it easier to integrate the two.

In collaboration with the Australian Defence Science and Technology Organisation (DSTO), we have used Java and CORBA to develop a prototype implementation of a system for accessing distributed active data archives of geospatial image data such as aerial photography and satellite data [7, 20].

The software is written using Java and CORBA, and conforms to a subset of the Geospatial and Imagery Access Services (GIAS) specification [39], which defines standard interfaces to a geospatial image archive using IDL (see section V for further information on interface standards).

The GIAS interfaces allow an application to to remotely invoke services such as storing and accessing image data and metadata; searching for images by querying the metadata; extracting image data for a specified region and converting the images between different formats; as well as general infrastructure to manage the whole system and handle requests and errors.

In our implementation, metadata was stored in a database and accessed via the Java Database Connectivity (JDBC) standard interface, while the images themselves was stored on the file server, for scalability and faster access and dissemination.

There are other distributed computing technologies, such as ActiveX and DCOM from Microsoft, but we do not consider them here since they are not portable or interoperable between different computing platforms, which is important for distributed systems.

XML [43] (eXtensible Markup Language) is becoming a common mechanism for specifying and delivering structured information for use in Web-based and distributed applications. It is particularly useful for developing systems for managing active data archives, since it can be used to provide a standard mechanism for describing everything from metadata schemas for database queries to the scheduling of processes across multiple compute servers for distributed processing of data. XML is gaining a lot of support as an enabling technology for Internet-based commercial transactions, and is also generating a lot of interest as a means of accessing geospatial [27, 28] and scientific [3] data archives. We have also developed an XML-based scripting language for specifying job scheduling for distributed applications [18].

Java and CORBA provide good support for developing applications that use on-line data archives, however the program developer must create a lot of software infrastructure, particularly to support multiple distributed data archives and the combining and scheduling of processing services that may be run across multiple distributed servers. A better approach is to use a higher-level *metacomputing* [15, 34] or *grid computing* [14] environment that provides high-performance middleware for handling distributed data storage, access and processing.

III. Middleware for Active Data Archives

The term middleware is now widely used to refer to software glue that hides the vagaries of networks and architecture dependence of distributed systems from applications codes. Middleware is generally thought of as that software which interfaces between the operating system controlling a particular computer and the application codes. Identifying a separate middleware component allows applications codes to be developed and maintained more portably and independent of a particular target computer system.

A complex scientific or decision support application may require access to multiple data archives and a variety of processing services, which may be distributed across a wide-area network. For rapid response, the application may require high-end compute, storage and network resources. These kinds of high-end applications, and the distributed active data archives that support them, can benefit greatly from a high-level middleware infrastructure that provides support for what is termed *metacomputing*, or large-scale computing on computing resources distributed across wide-area networks.

Metacomputing systems are mainly targeted at scientific simulations using distributed high-performance computers [14, 15, 34]. Most of the metacomputing systems currently in use do not utilise distributed computing technologies such as Java and CORBA that are becoming mainstream in the general information technology sector. Distributed Information Systems Control World (DISCWorld) [19] is our prototype metacomputing model and system, which has been developed using Java and builds on the modern object-oriented distributed computing models of Java and CORBA. It aims to provide middleware support for applications such as decision support systems that require access and processing of data from distributed data archives, particularly for large-scale applications that require large data sets and high-end computing platforms. It therefore provides a software infrastructure for supporting distributed active data archives, and the applications that use them.

A DISCWorld application example drawn from the area of image processing might involve an agricultural user query. A land policy manager is perhaps attempting to understand how drought is affecting a particular region. Understanding this issue might involve calling up satellite or aerial reconnaissance imagery on a particular region, perhaps overlaying or mosaicing several separate images from different passes of a plane or satellite. These images might represent different wavelength channels and it may be necessary to combine them to derive information about vegetation cover or soil/land cover type. A well integrated active archive would allow the user to call up sets of raw data such as the imagery; compose a regional filter that focusses on a geographic area of interest; invoke image mosaicing and registration operators that combine individual images into a composite covering the relevant region; invoke channel fusion operators that combine different remotely sensed wavelengths into a vegetation or greenness index for example; and call data reduction operators that can give area weighted percentages histograms or average values. The user query may involve moving around many gigabytes of imagery; carrying out mathematically and computationally intensive processing operations to composite images and channels; and finally powerful data reduction operations.

Almost all of the work involved can take place at the server side, with only relatively small thumbnail images or simple histograms and results being transmitted to the client. The server side may consist of a distributed set of compute servers, but these may be connected with high bandwidth networking technology, in contradistinction to the client connect which may be only a modem or terminal line. Making a clean separation between client and server side work, and successfully managing the scheduling or processes and data movements is a non-trivial task. Our DISCWorld middleware system was designed to address this.

DISCWorld is targeted at wide area systems and applications where it is worthwhile or necessary to run them over wide areas. These will typically be applications that require access to large specialist datasets stored by custodians at different geographically separated sites. An example application might be in land planning [5], where a client application may require access to land title information at one site, digital terrain map data at another, and air reconnaissance or satellite imagery stored at another site. A human decision maker may be seated at a low-end compute platform running a Web browser environment, but is able to pose queries and data processing transactions of a network of DISCWorld connected servers to extract decisions from these potentially very large datasets without having to download them to their own site.

Our vision for DISCWorld is an integrated query-based environment where users may connect to a "cloud" of distributed high-performance computing, communications and storage resources, and request data retrieval and processing operations. The user need not know anything about the details of the resources being used (which is why we use the cloud metaphor), since the DISCWorld middleware handles these. Users themselves may only be connected into the cloud by a low bandwidth network link such as that provided by a modem line. This action-at-a-distance query-based approach appears an appropriate one for decision support applications where the user is provided with a collection of application components that run in a Web browser and help the user to control remote high-end computing resources.

Much of our research to date has considered the multithreaded software daemon (DWd) that runs on each participating DISCWorld service providing host. This module is symmetric and provides the core functionality of both a DISCWorld server and a client. Servers can be clients of one another. Typically, a particular DISCWorld node will be configured to offer only a small set of services, which may be simply to act as a gateway to other services; to carry out storage management; to schedule processing tasks; or to act as a repository for code modules that can be loaded into a reconfigurable graphical client in a Web browser environment. DISCWorld provides a scheduling and brokering middleware for building active data archive servers. It interfaces to data storage management software, embodied as a separate software module which we describe in section IV.

The DISCWorld environment automatically assumes the responsibility of locating the data and program modules, and their transfer between distributed machines. We believe this approach is invaluable in the more general case of distributed

active digital archives, where the location and availability of data and program code is not assured, and large geographical separation of clients and servers makes all data transfer expensive. In addition, results are named in such a manner as to facilitate deferred delivery mechanisms. This allows the user to request some processing and then examine the results at a later time.

Figure 1 shows an interactive DISCWorld client interface that could run either as a Java application or Java applet inside a Web browser. A user will have logged on to the DISCWorld environment, using one of the participating server nodes as a gateway, and as shown has downloaded customisable imagery catalogue browser applets which can guide them through the data search process. The screendump shows that the user has already found two images which match the search criteria, and these have been returned as iconic pointers. These appear as thumbnail iconic forms of the found images. Figure 2 illustrates the subsequent display when the user has clicked one of the iconic pointer images and has downloaded it to the client.

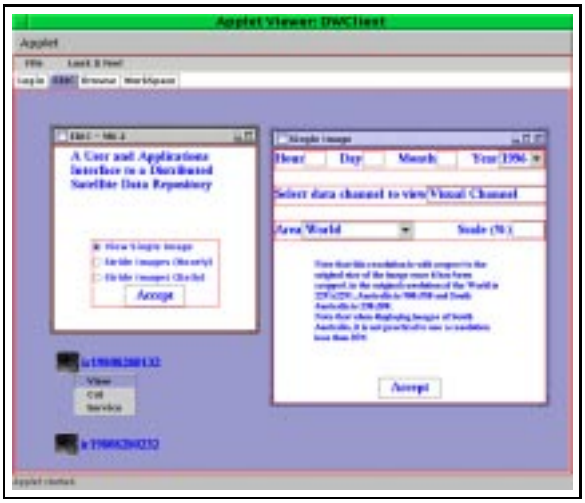

Fig. 1. Earth Observation Information Catalogue (ERIC) applets downloaded into a DISCWorld graphical client environment, running as a Java applet in a Web browser environment. The ERIC applet provides search forms to identify satellite imagery stored on a remote server, and the two iconic pointers at the lower left are remote data pointers to found data.

IV. Distributed Data Storage

On-line data archives place demanding requirements upon storage and network resources. Any underlying software infrastructure must therefore be able to optimise the storage and retrieval of data within and from these distributed and hierarchical resources to obtain high performance. In this section we outline the DISCWorld Storage System (DSS) and the DISCWorld File System (DWorFS) that underpin many of the operations of the DISCWorld environment.

The heterogeneous nature of distributed data archives implies that the underlying software infrastructure should be portable across host hardware and operating systems as well as storage mechanisms and network technologies. This issue is

Fig. 2. The user has invoked the download operation on one of the found images and has downloaded a copy of the data onto their display screen area. The data is now cached on the client and can be manipulated locally. The pointer to the server copy may still exist however, if the user has not explicitly discarded it.

complicated by the need for high performance, which is often at odds with the goal of portability. For example, experience has shown that methods often used to achieve maximum theoretical network performance are often non-trivial and platform specific.

The ever-increasing size of modern data archives implies that they are more likely to be both distributed (multiple servers connected across a network) and hierarchical (use tertiary storage such as tape silos, as well as disk arrays). Therefore any software infrastructure providing access to such archives must feature the ability to handle the underlying storage technologies and their potentially distributed nature. Increasing the importance of this distribution are the typical latencies across global networks, which are orders of magnitude greater than existing distributed data access mechanisms were designed to accommodate, and cannot be reduced due to lower bounds forced by the speed of light. Unlike most existing remote data access systems, future systems must be designed and implemented for latency tolerance. A similar problem occurs with hierarchical storage systems such as large robotic tape silos, which have access latencies which may be an orders of magnitude greater than those from disk. Essentially, the distributed and hierarchical nature of large-scale on-line data archives have presented equivalent problems – high-latency access to data held on differing, often proprietary storage technologies.

The large scale of modern on-line data archives also poses problems for the infrastructure developer in that the range of sizes of individual pieces of data within the system can be very large. For example, metadata objects can consist of just a few bytes each, yet a single multi-spectral satellite image can require hundreds of megabytes of storage. Systems designed to handle the storage and distribution of data within such archives must allocate, utilise and deallocate available resources in the most judicious manner, which can differ dras-

tically for differing sizes of data. For example, many network or distributed file systems [32, 33, 37] are aimed for use within "typical" operating environments by "typical" users. Studies [1, 29] and seemingly appropriate assumptions have been made regarding file size distributions, and designs tailored accordingly; however these assumptions may have no reasonable basis for large data archives. The same is true for both storage and networking resources. Maximising performance therefore relies on the ability of the software infrastructure to dynamically optimise the organisation of data across storage resources according to the data composition and performance characteristics of the given storage and networking resources.

Through our DISCWorld metacomputing project, we are exploring the storage facet of on-line data archives with the DISCWorld Storage Service (DSS) [31]. The DSS is a decentralised service layer within the metacomputing system, providing access to arbitrarily distributed and disparate storage resources. The primary design feature of the DSS is its modularity and extensibility. The DSS provides a framework for dynamically loaded modules that are specific to particular datasets or storage mechanisms to provide optimised access to on-line data archives and the underlying storage technologies.

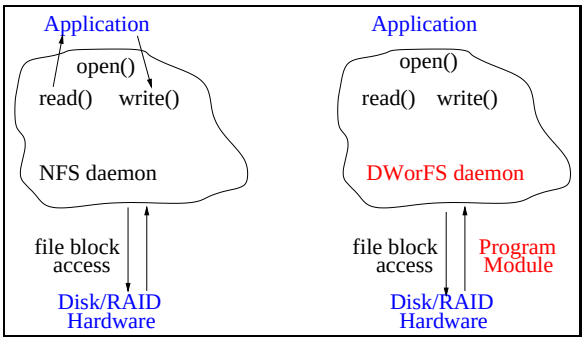

Fig. 3. DISCWorld File System compared to conventional Networked File System.

One of the building blocks for the DSS is the DISCWorld file system daemon (DWorFS) [30]. Network file systems operate by providing a software daemon that application programs interact with via systems calls such as *open(), read()* and *write()*. The daemon will transparently open files, and read or write data blocks to/from the actual disk system even if it is in fact controlled remotely by a networked computer. The standard Network File System (NFS) [37] daemon provides transparent access to the actual file system stored on a disk. Our DWorFS daemon is implemented as a plug in substitute for conventional NFS (see figure 3). It too provides applications with system call services so that the application appears to be accessing files as normal. DWorFS however is able to trigger program modules to intercept particular file access calls and take appropriate actions. For example, by choosing a naming convention for datasets that do not exist yet, but for which there is a known program or set of programs to run to create them, DWorFS can create data on the fly as and when it is requested by users. This is especially useful for the active data archive model we have described, whereby derived data sets can be produced from a static collection of primary data on demand. The derived data can be cached and the DWorFS module can interact with a cache management database to avoid reproducing data that already exists as a real file, left over from a previous request.

The DSS operates as a set of mutually co-operative server processes (using the DWorFS building block) on nodes within the DISCWorld metacomputing system, as illustrated in figure 4. No hierarchy exists between these servers; the distribution architecture is purely peer-peer. Requests received by a DSS server are forwarded to remote DSS servers or to lower levels within the server, which multiplex requests for dataset- and storage-specific modules. These modules can then provide optimised storage services for specific datasets or storage devices, for example satellite imagery archives and hierarchical storage mechanisms. Data access operations with disparate and arbitrarily distributed data archives and storage devices can then occur transparently without requiring client processes tp provide special support for such archives and devices.

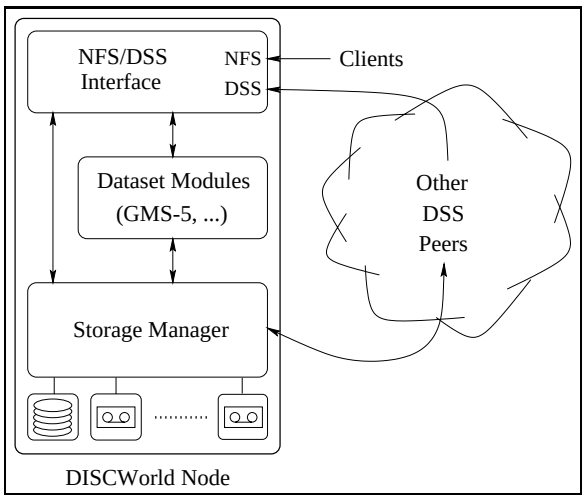

Fig. 4. Overview of the DISCWorld Storage Service architecture.

The modules within DSS are currently implemented through the dynamic loading facility available on most Unix-like operating systems. In response to a valid request received by DSS from clients or other DSS servers, the appropriate function of the specified module will be called to process the request. As executable code these modules can then of course perform arbitrary allowable operations to satisfy the request, including the generation and processing of data. We term this active nature of the storage system *active storage*, in that the modules providing access to storage and data resources can actively adapt to user requirements and resource availability.

The currently implemented interface to DSS is NFS. While not optimal for performance and other reasons, we chose to implement and utilise this interface as it allows access to the DSS infrastructure without requiring any application or operating system modifications. Each DSS node operates as a user-level daemon on a Unix-like operating system, waiting for NFS requests. Applications simply go through their system's local DSS mountpoint to access the DSS infrastructure. DSS then handles bulk remote data transfer, since NFS is not designed to do this, and will break large files into a lot of small chunks, which reduces performance. Incoming pathname re-

quests are processed by DSS and module(s) identified by DSS as responsible for the given directory subtree. For example, to access a GMS-5 satellite imagery archive through DSS, an example pathname request is:

```
/dss/GMS5/199901010930/vis/sa.hdf
```

Assuming `/dss` is the mountpoint for DSS, the `GMS5` pathname component is then interpreted as the dataset module to handle the request. The rest of the pathname, which looks like it points to a real file stored in a directory tree based on parameters or metadata fields specific to the dataset module, is actually a mechanism to specify to the DSS how to create the file based on these parameters. In the case of the GMS5 module, the format of the pathname is defined to specify the date and time, image spectral channel, location and file format respectively. The GMS5 module can then perform whatever operations are required to present the virtual file to the user application, including communicating with storage modules to access arbitrary local or remote storage resources and performing image transformations or conversions. The user application is completely oblivious of all this activity, and simply requests and accesses the data as if it were a normal file.

The DWorFS and DSS provide a simple but powerful way of accessing active data archives. They are particularly useful for legacy applications that can only access data through file requests. The extensible and active nature of the DSS allows almost arbitrary functionality to be implemented via modules. Through this, we plan to investigate a range of distributed and hierarchical data organisation and transfer paradigms applicable to on-line active data archives.

The DSS infrastructure allows us to integrate many different forms of storage management technology. For example, a client application program running on a PC can access a networked DSS mount point, which acts as an interface to a high capacity tape silo. The application issues normal file read and write calls, but the data it is accessing could be stored on tapes in a facility many thousands on kilometres away. The latency delays will of course significantly affect the application performance, but this is still a very useful technology for integrating together the storage access components of an active data archive.

V. Interface Standards for Active Data Archives

Providing on-line access to data archives enables any number of different applications to access data from a particular archive, and also allows a specific application to access data from multiple distributed archives. In order to support interoperability and easy access to multiple data archives from multiple application programs, it is crucial to define standardised interfaces to query, process and access data from on-line archives.

Many large on-line data sets consist of geospatial data, including vector data such as region boundaries, roads, and rivers; grid-based data such as elevation and land use; and raster image data such as aerial photography and satellite data. Spatial data is used in many different applications, and is often crucial for decision support systems.

It is therefore not surprising that there has been a lot of effort to provide metadata and interface standards for geospatial data, by a number of groups such as the Open GIS Consortium (OGC) [26], the ISO technical committee on geoinformatics (ISO TC211), the U.S. National Imagery and Mapping Association (NIMA), the U.S. Federal Geographic Data Committee (FGDC), the Committee for Earth Observation Satellites (CEOS), etc. The ultimate goal of these efforts is to enable a standardised global spatial data infrastructure.

Standards for querying data archives require standards for specifying metadata. A number of standards bodies have been tackling the issue of standard metadata fields for some time now, to allow clients to pose queries that will be understood by data archives that conform to the appropriate metadata standards. General metadata standards are being developed for searching Web sites and on-line digital libraries [10], but most of the effort involves the development of metadata standards that reflect the data content in particular fields of interest, for example the work by ISO, OGC, FGDC and CEOS to develop metadata for geospatial data. A problem here is that many data sets are used by different user groups that may define their own (incompatible) metadata standards or data models. A similar issue is how to handle revisions to the data model, so that if a server is upgraded to use new metadata fields, client applications that use a previous version will still work. A solution is to develop interfaces for querying metadata that can support multiple data models or data views.

More recently there has been a move towards standardising interfaces for accessing and processing data from on-line data archives, particularly for scientific data [20, 42]. NIMA and the Open GIS Consortium have been particularly pro-active in developing standard interfaces to spatial data archives for a variety of distributed computing technologies, including Java, CORBA, and Web interfaces.

Our research into software support for distributed archives of geospatial image data included the development of a Java and CORBA implementation [7] of NIMA's Geospatial and Imagery Access Services (GIAS) specification [39]. GIAS was the first widely-used standard for accessing geospatial data archives using object-oriented distributed computing, and a variant of this specification has recently been accepted as part of the OpenGIS Catalog Access Services standard [27], which supports both CORBA/IDL and HTTP/CGI interfaces. We also worked on improving the specification and implementation of the GIAS to allow more efficient access of multiple federated data archives [12].

Developing a software architecture and associated interface standards to enable efficient distributed *processing* of geospatial image data from on-line data archives is a much more complex problem than specifying interfaces for querying and accessing data. NIMA is developing the Geospatial and Imagery eXploitation Services (GIXS) framework [40] to address this problem, however it is currently much less mature than the GIAS specification, and a lot of work still needs to be done to create a comprehensive framework that is efficient and easy to use. Sun are also tackling this issue with their Java Advanced Imaging (JAI) [35] package. We are currently working with DSTO on an implementation of the GIXS using JAI, and investigating potential improvements to both of these specifications [8].

As the development and use of active data archives and dis-

tributed computing technologies grows, it is likely that standards for interfacing to on-line data archives will be developed for many different application areas and user communities. Since much of the basic functionality of these interfaces will be common to all application areas, it is to be hoped that organisations such as the Object Management Group (developers of CORBA standards), Sun (Java standards), and the Grid Forum [13] (standards for wide-area metacomputing or grid computing), can work towards developing general standards for querying, accessing and processing data from distributed on-line data archives.

VI. Summary and Future Work

Our experiences in building active data archive systems have allowed us to explore and evaluate many of the commonly available distributed computing technologies. We conclude that a wide-area on-line data archive, which will consist of many separately owned and managed data collections, requires:

1. A smart client, consisting of a lightweight core that can reconfigure itself from a networked repository of domain-specific interface components, and which will allow users to pose queries of the archive system;
2. A set of bulk and metadata management components, that will in practice consist of off-the-shelf components such as relational databases and large filesystems on disks, arrays of disks, and bulk data media such as robotic tape silos;
3. A flexible set of software components to allow storage technologies to be integrated together, under control of the middleware system;
4. A middleware system to allow data requests, transfers and processing actions to be invoked using standard interfaces.

We believe Java is currently the best software technology for 1. A mix of Java and CORBA technologies with close adherence to emerging standards for data access and processing is appropriate for 4. Storage systems will inevitably consist of semi-proprietary solutions such as databases and bulk data mangement systems, but we believe 3 can be usefully built from well designed systems level components such as DWorFS to interface to 4 and shield the systems developer from too many changes in 2.

There are a variety of distributed computing technologies that can be used to interface to on-line active data archives, and the best one to use will depend on the type of application. Many of the current interfaces to on-line data archives use basic Web technologies, accessing data from forms-based Web pages via HTTP and CGI. This is fine for applications accessing mostly static data from a single data archive, with perhaps a limited selection of processing services and a client that only requires a relatively simple, static interface. A more sophisticated, interactive interface that allows some processing on the client side can be achieved using a Java applet, which can still access data using HTTP and server-side processing services using CGI.

A more complex application may require access to multiple data archives, and a greater variety of processing services. This can be more readily achieved using Java and/or CORBA, which provide more advanced distributed and object-oriented programming environments. Many active data archives and applications, particularly with large distributed data sets and substantial computational requirements, can benefit greatly from a higher-level middleware infrastructure (such as DISCWorld) that provides metacomputing support to transparently and efficiently handle distributed data archives and high-end compute, storage and network resources.

There are a number of outstanding research issues to be properly addressed in constructing a reliable middleware system for supporting on-line active data archives. We believe we have a suitable development framework to usefully experiment with most of these. Our current system addresses many of the scheduling and process placement problems [24]. We have carried out some preliminary experiments in incorporating encryption and digital signatures in our system [16], and it now appears that many of these problems can be devolved to a careful deployment of the Java Cryptix package [38] in client and server side codes. There remain issues of encryption performance to be addressed however. Our DISCWorld middleware and storage system components are already fairly portable and could be implemented on almost any modern computer system. The middleware daemon, which is written in pure Java, is already bytecode portable.

The security of raw and derived data is one of the biggest concerns of creators and maintainers of large digital libraries. Closely related to the security issue is that of data economics and electronic commerce. Part of the security measures for a digital library is that users be held accountable for the data they access, or the processing services they use in an active data archive. This may take the form of a monetary amount, or perhaps limits on data sizes for downloads or CPU time for processing services. Also, different users may have different access privileges to sensitive data.

Incorporating security and electronic commerce transaction support to enable an on-line data archive to function in a commercial arena is an area we have not yet addressed. However technologies are emerging to allow secure data transmission and authenticated transactions support for electronic commerce activities, and Java based e-commerce systems are already being uptaken by financially demanding organisations such as banks. It is likely that the frameworks that are emerging will allow ready incorporation of e-commerce support into a Java-based system such as we describe.

The key problem of designing a scalable system using present technology is still unsolved. The peer-peer approach we adopt does not present any major bottlenecks, but scalability cannot be assured until a full-scale distributed system with realistic usage patterns is built and tested.

In conclusion, we believe there is at least a tractable route to addressing, if not completely solving, all the known problems of constructing a wide-area on-line data archive.

We are presently expanding the functionality of our middleware daemon and are developing a collection of reconfigurable client interface components. We continue to focus on areas involving bulk data and high-end computing, communications and storage, such as geospatial data and imagery access and analysis. This area has the additional advantage that some of the standardisation issues for data transport and interoperability are being addressed by the emerging OpenGIS standards, which offer an interesting opportunity for testing our

ideas and designs for building advanced active data archives.

Acknowledgments

This work was carried out under the On-Line Data Archives (OLDA) Program of the Advanced Computational Systems (ACSys) Cooperative Research Centre (CRC) and funded by the Research Data Networks (RDN) CRC. ACSys and RDN are funded by the Australian Commonwealth Government CRC Program. We thank James Hercus, Katrina Kerry, Kevin Maciunas, Jesudas Mathew, Andrew Silis, Duncan Grove and Francis Vaughan for their contributions to some of the OLDA projects described here.

References

[1] C.J. Antonelli and P. Honeyman. Integrating Mass Storage and File Systems. In *Twelfth IEEE Symposium on Mass Storage Systems*, pages 133–138, April 1993.

[2] BBN Technologies. OpenMap. http://openmap.bbn.com/.

[3] Kent Blackburn, Albert Lazzarini, Tom Prince, and Roy Williams. XSIL: Extensible Scientific Interchange Language. In *Proc. HPCN'99*, Amsterdam, April 1999.

[4] Yuh-Jye Chang, Paul Coddington, and Karlie Hutchens. Viewing the Visible Human using Java and the Web. In Y. Tang *et al.*, editor, *Web Technologies and Applications, Proc. of the Asia Pacific Web Conference (APWeb98)*. International Academic Publishers, 1998.

[5] P. D. Coddington, K. A. Hawick, and H. A. James. Web-Based Access to Distributed, High-Performance Geographic Information Systems for Decision Support. In *Proc. of Hawai'i Int. Conf. on System Sciences (HICSS-32)*, January 1999.

[6] P.D. Coddington, K.A. Hawick, and H.A. James. Web-based Access to Geospatial Image Archives. Technical Report DHPC-089, Distributed and High Performance Computing Group, University of Adelaide, June 2000.

[7] P.D. Coddington *et al.* Implementation of a Geospatial Imagery Digital Library using Java and CORBA. In *Proc. Technologies of Object-Oriented Languages and Systems Asia (TOOLS 27)*. IEEE, September 1998.

[8] P.D. Coddington *et al.* A Software Infrastructure for Federated Geospatial Image Exploitation Services. technical report DHPC-092, June 2000.

[9] Distributed and High Performance Computing Group. The NPAC/OLDA Visible Human Viewer. http://dhpc.adelaide.edu.au/projects/vishuman/.

[10] Dublin Core Working Group. Dublin Core Metadata. http://purl.org/metadata/dublin_core/.

[11] Robert Englander. *Developing Java Beans*. O'Reilly, 1997. ISBN 1-56592-289-1.

[12] M.W. Grigg *et al.* Component Based Architecture for a Distributed Imagery Library System. In *Proc. of the 6th Int. Conf. on Distributed Multimedia Systems (DMS '99)*, July 1999.

[13] The Grid Forum. Grid Forum Home Page. http://www.gridforum.org/.

[14] Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., 1999. ISBN 1-55860-475-8.

[15] Geoffrey C. Fox, Roy D. Williams, and Paul C. Messina. *Parallel Computing Works!* Morgan Kaufmann Publishers, Inc., 1994. ISBN 1-55860-253-4.

[16] Duncan A. Grove, Andrew J. Silis, J.A. Mathew, and K.A. Hawick. Secure Transmission of Portable Code Objects in a Metacomputing Environment. In *Proc. Int. Conf. Parallel and Distributed Processing Techniques and Applications (PDPTA'98)*, July 1998.

[17] Shishir Gundavaram. *CGI Scripting on the World Wide Web*. O'Reilly & Associates, 1996. ISBN 1-56592-168-2.

[18] K. A. Hawick and H. A. James. A Distributed Job Placement Language. Technical Report DHPC-070, Department of Computer Science, The University of Adelaide, May 1999.

[19] K. A. Hawick, H. A. James, A. J. Silis, D. A. Grove, K. E. Kerry, J. A. Mathew, P. D. Coddington, C. J. Patten, J. F. Hercus, and F. A. Vaughan. DISCWorld: An Environment for Service-Based Metacomputing. *Future Generation Computing Systems*, 15:623, 1998.

[20] K.A. Hawick and P.D. Coddington. Interfacing to Distributed Active Data Archives. *Future Generation Computer Systems*, 16:73, 1999.

[21] H. A. James and K. A. Hawick. Eric: A User and Applications Interface to a Distributed Satellite Data Repository. Technical Report DHPC-008, Department of Computer Science, The University of Adelaide, April 1997.

[22] H. A. James and K. A. Hawick. Resource Descriptions for Job Scheduling in DISCWorld. In *Proc. Integrated Data Environments Australia (IDEA5) Workshop*, 1998.

[23] H. A. James and K. A. Hawick. A Web-based Interface for On-Demand Processing of Satellite Imagery Archives. In Chris McDonald, editor, *Proc. 21st Australasian Computer Science Conf. ACSC'98*, volume 20 of *Australian Computer Science Communications*. Springer-Verlag, February 1998.

[24] H. A. James and K. A. Hawick. Scheduling Independent Tasks on Metacomputers. In *Proc. ISCA 12th Int. Conf. on Parallel and Distributed Computing Systems*, August 1999.

[25] Object Management Group. The Common Object Request Broker Architecture. http://www.omg.org/corba/.

[26] Open GIS Consortium. OGC Home page. http://www.opengis.org/.

[27] Open GIS Consortium. OpenGIS – Catalog Interface Implementation Specification. OpenGIS Project Document 99-051s, available from http://www.opengis.org/techno/specs.htm.

[28] Open GIS Consortium. OpenGIS Web Map Server Interface Implementation Specification. OpenGIS Project Document 00-028, available from http://www.opengis.org/techno/specs.htm, April 2000.

[29] J. Ousterhout, H.L. DaCosta, D. Harrison, J. Kunze, M. Kupfer, and J. Thompson. A Trace Driven Analysis of the Unix 4.2 BSD File System. In *Tenth ACM Symposium on Operating Systems Principles*, December 1985.

[30] C. J. Patten, F. A. Vaughan, K. A. Hawick, and A. L. Brown. DWorFS: File System Support for Legacy Applications in DISCWorld. In *Proc. Fifth IDEA Workshop*, February 1998.

[31] Craig J. Patten, K. A. Hawick, and J. F. Hercus. Towards a Scalable Metacomputing Storage Service. In *Proc. HPCN'99*, April 1999.

[32] M. Satyanarayanan *et al.* The ITC Distributed File System: Principles and Design. In *Proc. 10th ACM Symp. on Distributed Systems Principles*, volume 19, pages 35–50, December 1985.

[33] Alex Siegel, Kenneth Birman, and Keith Marzullo. Deceit: A Flexible Distributed File System. In *Proc. 1990 Summer USENIX Conf.*, pages 51–61, June 1990.

[34] Larry Smarr and Charles E. Catlett. Metacomputing. *Communications of the ACM*, 35(6):44–52, June 1992.

[35] Sun Microsystems. Java Advanced Imaging API White Paper (EA2). Available from http://java.sun.com/products/java-media/jai/, November 1998.

[36] Sun Microsystems. Java Technology Home Page. http://java.sun.com/.

[37] Sun Microsystems. Network File System Version 2 Specification. Request for Comments 1094, Network Working Group, March 1989.

[38] Systemics Ltd. Cryptix Home Page. Available from http://www.cryptix.org/, June 1999.

[39] U.S. National Imagery and Mapping Association. USIGS Geospatial and Imagery Access Services (GIAS) Specification, version 3.3, N0101-E. Available from http://164.214.2.59/sandi/arch/addinfo.html, June 1999.

[40] U.S. National Imagery and Mapping Association. USIGS Geospatial and Imagery Exploitation Services (GIXS) Specification, version 2.0. Available from http://164.214.2.59/sandi/arch/addinfo.html, June 1999.

[41] The U.S. National Library of Medicine. The Visible Human Project. http://www.nlm.nih.gov/research/visible/visible_human.html.

[42] Roy Williams ed. Special issue on Interfaces to Scientific Data Archives. In *Future Generation Computer Systems*, volume 16, 1999.

[43] World Wide Web Consortium. Extensible Markup Language (XML). Available from http://www.w3c.org/XML.