Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Parallel Computing 29 (2003) 1297–1333

PARALLEL COMPUTING

www.elsevier.com/locate/parco

Distributed frameworks and parallel algorithms for processing large-scale geographic data

Kenneth A. Hawick [a,*], P.D. Coddington [b,1], H.A. James [a]

[a] *Computer Science Division, School of Informatics, University of Wales, Bangor, North Wales LL57 1UT, UK*
[b] *Department of Computer Science, University of Adelaide, SA 5005, Australia*

Received 13 August 2002; accepted 23 April 2003

Abstract

The number of applications that require parallel and high-performance computing techniques has diminished in recent years due to to the continuing increase in power of PC, workstation and mono-processor systems. However, Geographic information systems (GIS) still provide a resource-hungry application domain that can make good use of parallel techniques. We describe our work with geographical systems for environmental and defence applications and some of the algorithms and techniques we have deployed to deliver high-performance prototype systems that can deal with large data sets. GIS applications are often run operationally as part of decision support systems with both a human interactive component as well as large scale batch or server-based components. Parallel computing technology embedded in a distributed system therefore provides an ideal and practical solution for multi-site organisations and especially government agencies who need to extract the best value from bulk geographic data.

We describe the distributed computing approaches we have used to integrate bulk data and metadata sources and the grid computing techniques we have used to embed parallel services in an operational infrastructure. We describe some of the parallel techniques we have used: for data assimilation; for image and map data processing; for data cluster analysis; and for data mining. We also discuss issues related to emerging standards for data exchange and design issues for integrating together data in a distributed ownership system. We include a historical

* Corresponding author.
E-mail addresses: k.a.hawick@massey.ac.nz (K.A. Hawick), paulc@cs.adelaide.edu.au (P.D. Coddington), heath@bangor.ac.uk (H.A. James).
[1] Tel.: +61-8-8303-4949; fax: +61-8-8303-4366.

0167-8191/$ - see front matter
doi:10.1016/j.parco.2003.04.001

review of our work in this area over the last decade which leads us to believe parallel computing will continue to play an important role in GIS. We speculate on algorithmic and systems issues for the future.

Keywords: Parallel computing; Distributed computing; Grid computing; Metacomputing; Geographic information systems

1. Introduction

Integrating parallel and distributed computer programs into a framework that can be easily used by applied scientists is a challenging problem. Such a framework has to enable easy access to both computationally complex operations and high-performance computing technologies, as well as providing a means for defining the appropriate data set for the operation. This paper presents an overview of our work over recent years on applying high-performance parallel and distributed computing to geographical information systems (GIS) and decision support systems that utilise geospatial data. Our main research interest is the development of metacomputing (or grid computing) systems that enable cooperative use of networked data storage and processing resources (including parallel resources) for large-scale data-intensive applications. Much of our work has addressed the issue of embedding high-performance parallel computing systems and applications in this kind of distributed computing framework. We make this transparent to the user by providing standard interfaces to distributed processing services that hide the details of the underlying implementation of the service.

We have focussed on GIS as an example application domain since it provides very large data sets that can benefit from parallel and distributed computing. For example, modern satellites produce on the order of 1 Terabyte of data per day, and there are many challenging problems in the storage, access and processing of such data. The enormous number and variety of geospatial data, metadata, data repositories and application programs also raises issues of interoperability and standards for interfacing to data and processing services, all of which are general problems in grid computing. Unlike a number of other fields, the GIS community has made significant advances in defining such standards. GIS also has a wide range of users and applications in many important areas, including oil and mineral exploration, forestry and agriculture, environmental monitoring and planning, government services, emergency services and defence.

Parallel techniques can be applied to geospatial data in a number of ways. Spatial data such as satellite imagery or raster-based map data lends itself well to a geometric decomposition among processors, and many applications will give good parallel speedups as long as the image size is large enough. For some applications on very large data, parallel processing provides the potential for superlinear speedup, by avoiding the costly overhead of paging to disk that may occur on a mono-processor machine with much less memory than a parallel computer. However, with the enor-

mous advances in processing speed and memory capacity of PCs in the past few years, the advantages of parallel computing for some applications are perhaps not as compelling as they once were.

Nevertheless, data sets available from some areas such as satellite imaging and aerial reconnaissance continue to increase in size, from improved spatial resolution, an increase in the number of frequency channels used (modern hyperspectral satellites may have hundreds of channels), and the ever-increasing sequence of images over time. There are still important opportunities to exploit parallel computing techniques in analyzing such large multi-dimensional data sets, as discussed in Section 3.

Analyzing this data might involve some interactive experiments in deciding upon spatial coordinate transformations, regions of interest, choice of spectral channels, and the image processing required. Subsequently a batch request might be made to a parallel computer (or perhaps a network of workstations or a computational grid) to extract an appropriate time sequence from the archive. On modern computer systems it is often not worthwhile spatially or geometrically decomposing the data for a single image across processors, however it may be very useful to allocate the processing of different images to different processors.

Not all spatial data is raster data and indeed many useful data sets will be stored as lists of objects such as spatial points for locations; vectors for roads or pathways; or other forms of data that can be referenced spatially. The nature of these lists of data mean that they are less obviously allocated for processing among processors in a load balanced fashion. Objects might be allocated in a round-robin fashion which will generally provide a reasonable load balance but may entail an increased communications overhead unless there are no spatial dependencies in the processing algorithms involved. An example that we have investigated is assimilating simple observational data into a weather forecasting system (see Section 4.1). The observed data from field stations, weather ships, buoys or other mobile devices can be treated at one level as an independent set of data that must be processed separately and which can therefore be randomly allocated among processors. However if such data must be incorporated onto a raster data set that is already allocated among processors spatially or according to some geometric scheme then a load imbalance may occur. The use of irregularly spaced observational data generally requires interpolation onto a regular grid structure, such as the spatial grid used for weather forecasting simulations or the pixels in a satellite image. This can be very computationally demanding, and we have explored some different parallel implementations of spatial data interpolation, which are discussed in Section 4.

Another type of GIS data is the metadata, which may or may not be spatial in nature. Metadata may be textual or other properties that are directly related to a spatial point or vector, but may also be associated with a spatial region or with some other subset of the bulk data. The processing implications for such data are similar to those for other list-based data with the added complication that metadata may relate to many other elements of list or raster data and associated parallel load balancing is therefore difficult to predict.

Data mining is a technique that is gaining increasing attention from businesses. The techniques generally involve carrying out correlation analyses on customer or

product sales data. The techniques are particularly valuable when applied to spatially organised data and can help in planning and decision support for sales organisations and even product development. Generally customer or sales data will be organised primarily as lists of data objects with associated spatial properties. Processing this data mostly lends itself to the independent decomposition into sets of task for processors or possibly presorting objects by spatial coordinates first. This is a promising area for development and exploitation of new and existing data mining algorithms [64].

Many GIS applications are interactive, real-time or time-critical systems. For example, spatial data and spatial information systems play a key role in many time-critical decision support systems, including emergency services response and military command and control systems. Parallel computing can be used to reduce solution times for computationally intensive components of these systems, and to help gather and process the large amounts of data from many different sources that are typically used in such systems. For example, environmental studies may require the analysis of data from multiple satellites, aerial photography, vector data showing land ownership boundaries, and other spatially referenced data on land tenure, land use, soil and vegetation types and species distributions. This data is typically owned by many different organisations, stored on different servers in different physical locations. It is often not useful or feasible to take local copies of all the data sets, since they may be too large and/or dynamically updated.

There has been substantial progress in the past few years on the concept of a *spatial data infrastructure* that would enable users of decision support systems or GIS applications to easily search and access data from distributed online data archives. This work has mainly involved the development of standard data formats and metadata models, as well as more recent efforts by the U.S. National Imagery and Mapping Association (NIMA) [48] and the OpenGIS Consortium [52] in developing standard interfaces for locating and accessing geospatial data from digital libraries [50,53]. One of our projects involves developing software to provide these standard interfaces for accessing distributed archives of geospatial imagery data [10,11].

We have also been investigating frameworks for supporting distributed data processing, as well as distributed data access (see Section 2). We believe these should be addressed in concert. Particularly for large data sets, it may be much more efficient for data to be processed remotely, preferably on a high-performance compute server with a high-speed network connection to the data servers proving input to the processing services, than for a user to download multiple large files over a lower bandwidth connection for processing on their less powerful workstation. However the user should not be burdened with having to make the decisions on finding the "best" data servers (the required data set may be replicated at multiple sites) or compute servers (there may be many different servers offering the same processing services). This is a difficult problem that should be handled by grid computing middleware including resource brokers and job schedulers.

The goal of the OpenGIS Consortium is to also provide standard interfaces for geospatial data processing services, in addition to the existing standards for search-

ing and accessing spatial data. However this is a much more difficult problem and there has been little progress thus far. NIMA has been developing the Geospatial Imagery eXploitation Services (GIXS) standard [49], however this is restricted to processing of raster image data and still has several shortcomings (see Section 5).

In a distributed computing framework with well-defined standard interfaces to data access and processing services, an application can invoke a remote processing service without necessarily knowing how it is implemented. This allows faster, parallel implementations of services to be provided transparently. A distributed framework could also provide concurrency across multiple servers, either in concurrent (or pipelined) processing of different services as part of a larger task, or repeating the same computation for multiple input data (e.g. satellite images at different times or places).

In the past few years there has been significant progress in the area of metacomputing, or grid computing, which aims to provide coordinated access to computational resources distributed over wide-area networks. Our work is aiming to develop high-level metacomputing middleware that will make grid computing easier to use, particularly for GIS developers and users. We envision a system whereby environmental scientists or value-adders will be able to utilise a toolset as part of a metacomputing/grid system to prepare derived data products such as the rainfall/vegetation correlations described in Section 4.2. End users such as land managers will be able to access these products, which may be produced regularly in a batch procedure or even requested interactively from a low-bandwidth Web browser platform such as a personal computer. Web protocols will be used to specify a complex service request and the system will respond by scheduling parallel or high-performance computing (HPC) and data storage resources to prepare the desired product prior to delivery of the condensed result in a form suitable for decision support. HPC resources will be interconnected with high-performance networks, whereas the end user need not be. In summary the end user will be able to invoke actions at a distance on remote data.

In this paper we present some of our work on applying high-performance distributed and parallel processing for applications that use large geospatial data sets. The paper is structured as follows. Section 2 provides an overview of our work on metacomputing and grid computing, particularly the design and development of distributed computing frameworks for supporting GIS services and applications, and describes how this type of framework allows transparent access to parallel implementations of these services and applications. Some examples of parallel GIS applications are given in Sections 3 and 4, along with some discussion of how they can be embedded in a distributed computing framework. Section 3 describes some image processing applications on large satellite images, that can be parallelised using standard geometric decomposition techniques. These include image filtering, georectification and image classification. For the case of image classification, we give an example of a Java applet that could be used to provide a front end user interface to a remote high-performance compute server, which may be a parallel machine or a computational grid. Section 4 presents two different approaches to the problem of developing parallel algorithms for spatial data interpolation, which involves

mapping irregularly spaced data points onto a regular grid. We describe how parallel implementations of this computationally intensive application can be utilised within a larger application, such as weather forecasting or rainfall analysis and prediction, and how this can be done using the DISCWorld metacomputing/grid framework.

The remainder of the paper discusses software architectures for distributed computing frameworks to support distributed GIS applications. Section 5 outlines some work on implementing a distributed framework for geospatial image analysis that is based on the proposed GIXS standard from NIMA. We discuss several key issues, including mechanisms for specifying the components of the distributed application and their interaction, the exploitation of concurrency in the distributed framework, and an analysis of the the proposed interface standards. We also describe an example application for target detection. In Section 6, we discuss our vision for the application of grid computing technologies to distributed geographical information systems and spatial data infrastructure. We present an overview of evolving grid protocols, services and technologies to support access and processing of large distributed data sets, and outline some of our work on developing prototypes of these virtual data grids for spatial data. We also discuss the important issue of standards for spatial data grids, which must integrate general standards for grid computing with standard interfaces for geospatial data being developed by the OpenGIS consortium. We conclude with a summary of our work and a discussion of what we see as some of the future issues in developing high-performance parallel and distributed applications for processing large-scale geospatial data.

2. Distributed computing frameworks and computational grids

In this section we describe some general issues relating to grid frameworks and focus in on the parallel and concurrency needs of a middleware system, then on specific GIS issues, and finally embedded parallelism.

A grid computing or metacomputing environment allows applications to utilise multiple computational resources that may be distributed over a wide-area network [19]. Each resource is typically a high-performance computer or a cluster of workstations, however the term encompasses other resources including storage facilities such as disk arrays or tape silos, data archives such as repositories of satellite data, data visualisation, and even computer-controlled scientific instruments.

Until very recently, most grid computing applications were in the areas of computational science and engineering that have traditionally been major users of high-performance computing. However as grid computing technologies become more sophisticated and easier to use, a much wider variety of applications is likely. GIS applications are an obvious candidate for grid computing, since they often require access to multiple distributed databases and large data sets such as satellite imagery, as well as significant computational resources for data processing and simulation, particularly for time-critical applications such as emergency services decision support and military command and control.

Although grid computing has enormous potential for a variety of applications, it is still "bleeding-edge" technology. Current grid computing software such as the Globus Grid Toolkit [21] (which has become a de facto standard for grid computing) is rather low-level, and is still difficult to set up, cumbersome to use, and requires substantial technical expertise.

Our work aims to make it easier to develop computational grid applications, by developing high-level software and services for data management, discovery, access and processing that are easy to use, provide enhanced functionality, and hide the low-level details from application developers and end users.

In particular, we aim to create high-level tools for developing and using spatial data grids, since we believe grid computing ideas and technologies can potentially be of great benefit for the development of distributed geospatial information systems and spatial data infrastructure, however little research and development has been done in this area.

Our On-Line Data Archives (OLDA) project [14,27] developed prototype computational grids consisting of large-scale distributed data archives, distributed computational resources including parallel supercomputers and networks of workstations, and our DISCWorld [29] metacomputing software to allow applications to exploit these distributed resources. Our focus was on GIS applications using large satellite data sets, and we constructed several prototype systems, including prototype application demonstrators for the Australian Defence Science Technology Organization (DSTO) and some South Australian Government departments.

2.1. Metacomputing middleware

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 interface. 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.

Metacomputing systems are mainly targeted at scientific simulations using distributed high-performance computers [19,20,59]. Many of the most prominent metacomputing systems currently in use, such as Globus, do not utilise distributed computing technologies such as Java and CORBA that have become mainstream in the general information technology sector. This situation is changing only slightly with some systems now offering mechanisms to interface with Java and CORBA, such as the Globus Commodity Grid Toolkit (CoG) [22]. A recent development is the emergence of grid standards such as the Open Grid Services Architecture (OGSA) [8] which are expected to promote interoperability and general wider acceptance of grid systems.

Distributed Information Systems Control World (DISCWorld) [29] 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 [26,27] that provide services for data access and processing, and the applications that use them. Current metacomputing systems are focussed on computation rather than data, although it can be argued that most applications that require grid computing are data-centric [58]. DISCWorld specifically focusses on providing high-level metacomputing middleware for distributed data processing.

The basic unit of execution in the DISCWorld is a *service*. Services are pre-written pieces of Java software that adhere to a specific API, or legacy code that has been provided with a Java wrapper. Use of parallelism is transparent, hidden behind a standard interface to a service. DISCWorld (and indeed any middleware system) needs to incorporate parallel computing and concurrency ideas in order to successfully integrate disparate systems components including massively parallel resources. We give some details of our DISCWorld system since they closely mirror ideas and recent developments in grid systems based on embedded parallel services.

The key to a working system lies in: the software design of the DISCWorld servers; the protocols used to communicate amongst themselves; and the interoperability mechanisms with legacy applications and data systems. We have approached these problems by placing a DISCWorld daemon on every server that participates in a DISCWorld (grid) community. The software for the daemon is intended to be a minimal framework for loading up customised modules for specialist services. The design is motivated by a need for scalability to a large number of cooperating nodes, with as little central decision making as possible. The local administrators or owners of a resource participating in a DISCWorld can decide what services they will offer by setting various policy and environment information in its local database.

DISCWorld acts essentially as a software glue, providing interoperability between new and existing services. Fig. 1 illustrates the key ideas behind the DISCWorld daemon "DWd". A dual network is present between all servers in principle. A control network is used to communicate small lightweight messages that specify how servers are interacting and cooperating, and a bulk data transfer mechanism is used to transfer data products and partial products between nodes. In practice these mechanisms may be implemented on the same hardware network, or on separate networks when available. The control information is lighter but needs a reliable network, whereas the bulk data needs high bandwidth but can always be retransmitted if a network is blocked. Daemons are used to provide all the necessary control information to wrap up legacy codes and systems as well as being a framework for interfacing to new codes. We have also built a file-system based mechanism to provide the support services necessary to support legacy applications for which we do not have source

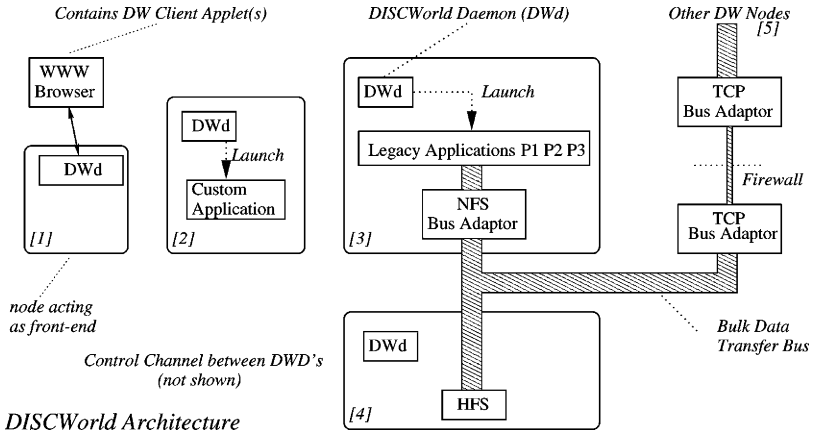

Fig. 1. The DISCWorld software architecture overview.

code access [55]. Software adapters are used to bridge firewalls as well as interface to other metacomputing software systems.

The daemon's operations are best explained by a discussion of the sequence of events that take place when a client contacts it.

- A user query is posed to the daemon (either through a graphical or text-based front end), in the form of a well-defined request for some computation to be performed on some data.
- The request must be mapped onto the available resources (operators mapped to particular compute systems) that the daemon knows about and even though the daemons are peer-based, the daemon in question acts as a manager for the purpose of brokering this request.
- Management of the request consists of monitoring and supervising the ongoing request once it has been initiated. The daemon must also provide the user with an ability to revoke the request should it be necessary.
- The daemon is also responsible for brokering the safe delivery of the results to the user, or temporary storage of the results and notification of the user. It can also organise the collection and storage of performance data for the purposes of predicting future performance and costing.

The daemon is initiated as a program by the system administrator or as a system daemon and, once initiated, it reads some local state information from a persistent store (e.g. database) to configure itself and subsequently listens to a particular port for user requests. Each daemon has a "gossip" mechanism to allow it to determine what services and hosts are available to it. Each daemon stores its own internal database of servers offering component services as well as their associated characteristics, and maintains a priorities list according to some locally defined policy. The daemon is highly configurable, and can be customised to reflect the local

server/services it provides to other DISCWorld daemons. It also has a configurable policy for where to seek assistance from its peers in satisfying incoming user queries.

DISCWorld was primarily motivated to deal with relatively large computations that need to carried out on large data sets. This is fundamental to sharing resources, as the items to be shared are quite often very large, for example, satellite imagery, or time-lapse sequences of imagery showing trends and interesting phenomena. The major services/servers that each daemon knows about were originally substantive programs running on high-performance platforms. Nevertheless, the control information and "glueware" needed to build such a system are complex but not so demanding of computational performance. DISCWorld is written in pure Java; whereas up until now we have mainly implemented operators as Java wrappers around high-performance legacy code, the rapid improvement in Java Virtual Machine performance has prompted us to build many more component services in pure Java. A Java implementation allows for a number of capabilities that can be readily implemented using the Remote Method Invocation (RMI) Java package. We have investigated some critical benchmark operations such as numerical analysis and image processing operations in pure Java and as native methods on various supercomputer platforms which can act as performance accelerators or specialist compute servers in a DISCWorld environment [43].

The multi-threaded mechanisms in Java allow the daemon to be implemented with a prioritised set of threads to handle the user and server–server interactions. The use of Java threadgroups coupled with the Java security manager mechanism enables threads to be placed into groups with controlled access between the groups. This enables the daemon control threads and groups of user threads to be kept apart and scheduled separately within the daemon. The daemons communicate via typed messages. These are received at a central point within each daemon, allowing messages to be routed to the appropriate module. Some of these modules within our current daemon prototype, include a server/services database manager, a system administration console manager, a computation organiser and a local execution manager. These only accept control level messages from the central point, although the DISCWorld model allows for a high bandwidth direct communication channel between modules. Control messages contain instructions for a single module, and if modules within the same daemon wish to communicate they must sent messages via the central point.

The use of the Java object serialisation and dynamic loading and binding allows arbitrary data structures to be passed between both servers and clients in the DISCWorld. The code for these data structures may be loaded dynamically. This includes operations on the data structure which means that the same data structure can have different method implementations depending on the data contained within—thus allowing polymorphic operations. At present, serialisation is limited by the need for both communicating parties to use Java, which requires Java wrappers around non-Java code. Wrappers can be done using the Java Native Interface (JNI) API. We believe the additional effort of implementing Java wrappers is outweighed by the ease with which objects with state can be stored and communicated.

Using Java coupled with native methods to embed other programs does not mean that the system itself is no longer portable. The glue that controls the system remains portable although the native method services will not be portable. This is in keeping with our intention that a particular daemon should be configurable by the local administrator to reflect the locally available server/services. In fact, a limited portability can be achieved by careful loading of the native library for some applications. The library that can be loaded can be dependent of the architecture and operating system version, which information can be made available to the daemon as part of its configuration. This is not completely dependable, however, and it is an attractive possibility to find a mechanism whereby a growing collection of application components can be maintained as portable modules. This concept was further expanded upon by using dynamic class-loading in the subsidiary Java Message Passing (JUMP) project [44].

The Java Bean mechanism is a useful way to approach this. Beans are essentially codes conforming to a strict application programming interface (API) through which the system can manipulate them. Native methods as well as Java code and also data can be encapsulated as Java Beans. We use the Java Bean mechanism to implement our portable code modules which are saved on a codeserver [45] and are transmitted as high-level instructions to servers at the request of clients.

The DISCWorld Remote Access Mechanism (DRAM) [38] provides a rich pointer to allow references between operators and data in the running system. This is particularly important for long running computations or in the case of partial failure of a system due to network vagaries or indeed some users going offline deliberately, such as reaching the end of their work shift. The DRAM facilities and its associated mechanism for data futures (DRAMFs) [35] provide a general architectural model for specifying how the system behaves under reconfiguration.

DISCWorld therefore provides us with a base platform architecture upon which large-scale imagery analysis tools can be constructed. We believe the combination of DRAMs, pointing to data, services and not-yet-created futures, together with an appropriate scripting language [30] provide the necessary framework to enable the efficient construction and utilisation of software to properly exploit distributed data sets and users in a wide-area distributed system.

2.2. Distributed GIS frameworks

The enormous growth in the use of the Internet has led to the realisation that powerful geospatial information systems and decision support systems could be developed that utilise multiple distributed spatial data sets. However the development of this spatial data infrastructure has been slow, primarily due to interoperability problems, since GIS programs were originally designed as stand-alone systems. The spatial data community has been attempting to address these interoperability issues. Initially this was confined to the development of standards for data formats and metadata schemas, but recently this has been extended to the development of standard interfaces for spatial data query, access and (to a lesser extent) processing services. The development of standard interfaces and protocols for spatial data

services is being handled by the OpenGIS Consortium (OGC) [52], which is made up of key GIS developers and users in industry, universities, government, and defence. Separate OpenGIS interface standards are defined for the most commonly used distributed computing technologies such as CORBA, COM, Java, and the Web.

Our research into software support for distributed geospatial data access and processing includes the development of a system for accessing distributed active data archives of geospatial image data such as aerial photography and satellite data [10,11,26], in collaboration with the Australian Defence Science and Technology Organisation (DSTO). The On-Line Geospatial imagery Access Services (OLGAS) software is written using Java and CORBA, and conforms to a subset of NIMA's Geospatial and Imagery Access Services (GIAS) specification [50], which defines standard interfaces to a geospatial image archive using Interface Definition Language (IDL).

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 [53], which supports both CORBA/IDL and HTTP/CGI interfaces. The GIAS interfaces allow an application 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.

We have also worked on improving the specification and implementation of the GIAS to allow more efficient access of multiple federated data archives [24,33], for example by providing federated queries across multiple libraries, and if multiple copies of a desired data item exist, analyzing network performance to each library, and identifying and using the library that will provide the fastest download time for the image data. The standard interfaces currently provide no support for services across multiple federated data archives.

The OLGAS software provides a framework for accessing geospatial image data from distributed repositories, however it has very limited support for geospatial data processing. Metacomputing frameworks such as DISCWorld [29] and PAGIS [66] provide support for the specification and execution of arbitrary distributed processing. Our goal is to develop simple but powerful interfaces that allow users to easily specify a complex task based on a set of known services, and have the metacomputing middleware be responsible for finding, scheduling and coordinating distributed resources to carry out the desired task.

We are developing some example applications using the prototype DISCWorld metacomputing middleware. These include the rainfall analysis and prediction application described in Section 4.2. We have also developed a prototype framework for distributed processing of geospatial imagery that uses CORBA and the Java Advanced Imaging (JAI) [61]) API, with interfaces conforming to NIMA's Geospatial Imagery eXploitation Services (GIXS) standard. This work is described in more detail in Section 5.

2.3. Transparent embedded parallelism

Despite recent success in mainstreaming parallel processing it is often off-putting to an applications user to have to be aware of the parallel engine in their system. The computational grid approach provides a very useful way of embedding parallel computer systems or other special purpose processing systems in a way that is transparent to the everyday user.

It is feasible and useful to take an applications front end such as provided by a commercial GIS or by another problem solving environment (PSE) such as Matlab and use remote parallel computer servers to significantly reduce the solution time for computationally intensive services provided by the GIS or PSE [56]. Successful prototypes of this approach in the area of numerical linear algebra include Mathserver [56] and Netsolve [7]. Our early prototype GIS environment ERIC [36,37] and many of our targeted applications of DISCWorld have used this approach to embed parallelism in a grid-like infrastructure.

The ERIC system employed Web client/server technology to allow users to access and process satellite imagery using a Web browser interface to a CGI program running on a Web server. A number of distributed and parallel processing technologies can be embedded in this infrastructure. We use remote execution in the form of distributed `rsh` invocations to farm out parts of the processing to a cluster of workstations. Parallel programs running as remote services either on the workstation farm or on a dedicated machine are also exploited. For example, when browsing a sequence of images it is useful to create a thumbnail resolution time sequence of images that can be played through the Web interface as a movie sequence. We integrated a parallel MPEG encoder into our infrastructure to provide rapid interactive access to movie sequences of specified spectral channels, spatial regions and time sequences.

As part of the components technology investigation for the DISCWorld project, we have investigated the use of a networked parallel computer as a suitable accelerator platform for carrying out computationally intensive operations such as Kriging spatial data interpolation, described in Section 4. The parallel computer runs as a server and provides the computational services to client applications running on networked lightweight computing platforms.

Interesting issues in the construction of such a system are the tradeoffs between the communications cost of transferring data from client and server compared to the speed advantages of using a powerful computational server instead of running the computation on the client. In the case of Kriging we are able to analyse the problem complexity in terms of the problem size specified by the number of input and output data set points. The parallelism for dense matrix problems is well understood so it is possible to parameterise the problem scaling with different numbers of processors allocated [8].

It is also possible to allow the client application to choose which computational accelerator for the task required. There may be a real monetary cost associated between fast, medium and slow servers all of which are functionally capable of providing the same service. Other issues include network reachability of the servers and estimated time to deliver the solution compared to the end user requirements. The

problem complexity also varies as other pre-processing and post-processing activities are required by the end user. The cost tradeoff space can be explored using relatively simple linear models and stored coefficients from previous problem runs.

Parallel computational modules in our system are integrated together using a client–server model. While it is possible to implement a simple system using communicating Java applets and code that runs on the Web browser [7], a more powerful system links together high-performance computing resources to carry out computationally intensive tasks [57]. Of particular interest to us is chaining processing requests together so that all data intensive calculations can be performed remotely. This model of computation is often referred to as workflow or dataflow. We have built a Java front end to individual processing modules which are each managed by a Java server and which implement the workflow paradigm.

We are investigating ways to allow servers to communicate together peer-to-peer to satisfy a complex user processing request. For example, a catalog server might respond to a user request to specify a sequence of images to extract from the repository. The metadata specifying this sequence might be passed to the repository storage manager server which would feed the images to a temporary staging area. The user might have specified some processing to be applied to the entire sequence, for example image classification using the classification applet described in Section 3.3. The classification matrix can then be applied to the entire sequence of images by a classification server engine. Finally, the resulting artificially coloured images might be merged together into a movie sequence by a different server. We are investigating protocols for servers to be managed in a hierarchy, with user requests being serviced through an interface server.

The embedded parallelism approach hides the details of the parallel implementation and execution from the application and the user, which only see a standard interface to a computational service in a distributed computing framework. This approach relies on the definition of interface standards for well-known services, such as the interfaces for GIS being developed by the OpenGIS Consortium. The use of parallel computers could be completely transparent to the user, or the user could be offered a choice of servers with different performance capabilities, with a different cost associated with each. The cost could be monetary or some other measure, for example a user may have access to any number of cycles on a single node computer server, but have an allocation of a specified number of CPU-hours on a large parallel machine.

3. Satellite image processing applications

In this section we describe some image processing and analysis applications that we have developed for use as on-demand services that can provide input for GIS applications for various value-adders and end users. All these applications require significant computation, and can be computed in parallel to provide fast results, which is particularly useful for interactive or real-time systems. These applications were implemented for satellite image data collected from the Japanese GMS-5 satellite, however the techniques are generally applicable to any geospatial image data.

3.1. GMS-5 satellite data archive

The primary geospatial data set that we have used for our experiments in parallel and grid computing is an archive of image data from the Geostationary Meteorological Satellite (GMS-5) [23] deployed by the Japanese Meteorological Agency (NASDA). The archive contains all images from the past year stored on a RAID disk array, and previous years' data stored on a tape silo.

The Visual and InfraRed Spin Scan Radiometer (VISSR) [23] on the GMS-5 satellite produces a set of multi-channel images once an hour. The different channels represent different spectral frequencies, with one image in the visible part of the spectrum and the other three in the infrared range, including one designed to detect water vapour. Each image is a full-disk photograph of the Earth from an altitude of 35,800 km and a position almost directly over the equator at 140 degrees east (close to Adelaide's longitude). Examples of the images produced are shown in Fig. 2. The images are 2291×2291 pixels, with approximately 3 km pixel edge resolution. These seemed large images when we started this work although handheld digital cameras are now capable of this resolution. Satellite image and remote sensing resolutions have increased correspondingly however and $10{,}000 \times 10{,}000$ pixel images are not uncommon.

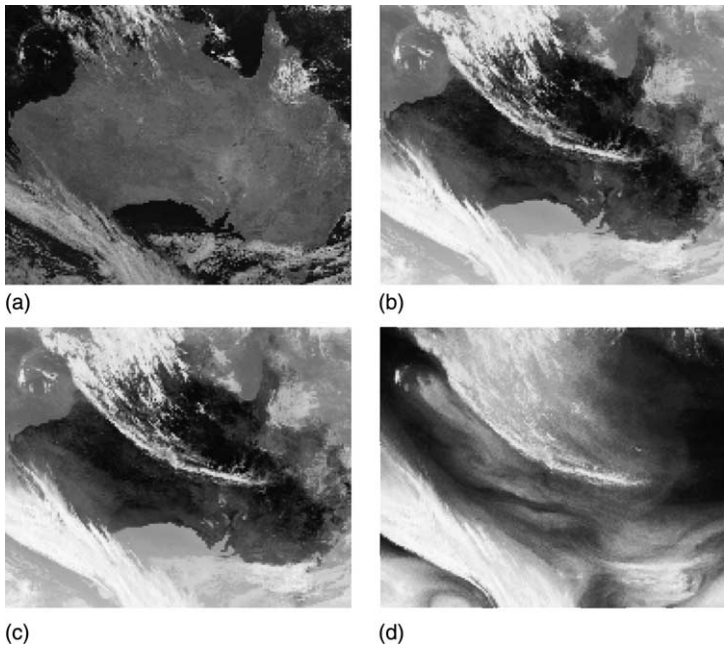

Fig. 2. Multi-spectral data cropped from GMS-5 worldview data: (a) visible spectra, (b) thermal IR1 spectra, (c) thermal IR2 spectra and (d) water vapour IR spectra.

Files are stored in Hierarchical Data Format (HDF), which stores the imagery as well as metadata. The HDF file format is supported by a library of utilities for extracting and manipulating the images and metadata, which we have integrated into our software infrastructure for accessing and processing the GMS-5 data. More details on our satellite data archive can be found in Ref. [28].

3.2. Image filtering and scanline corrections

The GMS satellite imagery is generated as a set of scanlines. Occasionally a scanline dropout occurs, for example from transmission errors, and a black or empty image line appears. In the case of single dropout lines it is possible to interpolate lines above and below to acceptable accuracy for some information products. A more robust approach is required to correct multiple dropout line errors. A useful technique is to Fourier Filter the image, and apply a wedge filter to remove artifacts in the spectrum that correspond to the dropped lines. This can be used to generate approximations to the dropped lines based on neighbouring values. We are experimenting with various parallel algorithms for applying Fast Fourier Transforms (FFTs) to what are (still relatively) large image sizes (2291×2291 pixels). These sizes are also inconveniently non-powers of two which requires padding or truncating the images. We are also investigating non-radix-two FFT algorithms for this purpose. However, modern computer systems typically have large amounts of memory available and no longer need necessarily employ memory conservative algorithms. This is particularly so for grid based systems where memory constraints are even less important.

3.3. Image classification

Processing image or raster data often involves simple nearest neighbour and stencil operations [39], however this is not always the case. A common form of analysis for multi-channel data is to histogram the data and identify clusters in the channel space that relate to specific features such as land or clouds. This is done by creating correlation plots between pixel values in different channels of data. These plots show the frequency of a pixel value set occurring in each channel. For example, as the wavelengths of the IR1 and IR2 bands in the GMS-5 images are very close, the correlation between the pixel values represented in the images is very high. Conversely, the correlation between the Visual and other channels is not high, and is somewhat localised, as shown in Fig. 3(c).

It is desirable that this type of image classification be carried out interactively, since there is a high degree of decision making necessary in choosing cluster radii parameters for the analysis. We have built an example system using a Java application to allow a user to classify and colour regions occurring in multiple channels of an image set. Once such parameters are chosen by the analyst, a batch job can be composed and submitted to a computer server to process a time sequence of data from the archive. The image classifier can be run in parallel using geometric decomposition of the images, or across different images in a time series.

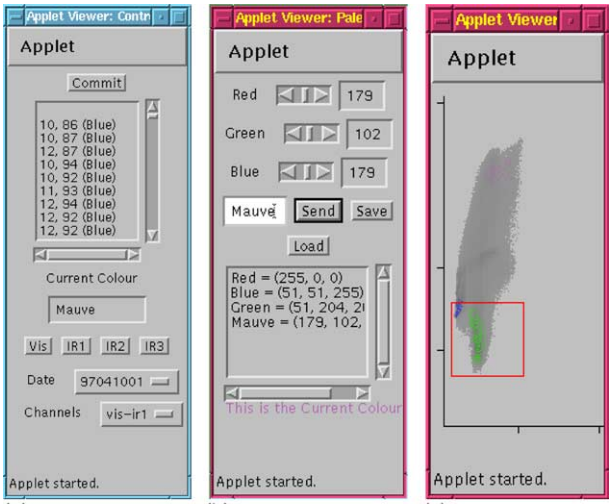

Fig. 3. Java image classifier: (a) control panel, (b) colour palette and (c) main histogram showing visible channel data (0–255) plotted vs infrared channel 1 (0–63), dark regions showing most common values.

By use of the control panel, as shown in Fig. 3(a), the user can select the data set with which to work, the images within that set to use for the correlation graph, and the image on which to superimpose the pixel colouring. The application loads a set of images, displaying the image selected in the control panel, and produces correlation plots of the selected channel's pixel values. These are displayed in a histogram such that the frequency of the pixel set occurrence is indicated by the darkness of the boxes, as shown in Figs. 3(c) and 4(a).

The colour palette, shown in Fig. 3(b), allows the user to select a colour that will be used to highlight the region of interest. The user is presented with a zoomed histogram, Fig. 4(a), in which they can select individual pixels. The action of selecting pixels in the zoomed histogram causes all pixels corresponding to that location to be changed to the current colour in the image viewer (Fig. 4(b)).

A list of the pixel sets that have been modified is stored; this list is recorded in a file, which may be used as input to a system that will automatically colour the pixels across different image sets. This automatic colouring will enable us to produce animated sequences of images, allowing visualisation of weather patterns as they progress in time, as well as other useful classifications.

It is possible, with some degree of accuracy, to derive the correlation of sub-images. For example, high-altitude cloud and ground have quite different correlation graphs. If the correlation graph of high-altitude cloud is known in advance, and is selected in the correlation graph of the whole image, this should result in the selection of all the high-altitude cloud for the image. If this is done for all the major components of the image, ie sea, high- and low-altitude cloud, and land, then it should be possible to almost fully classify and colour all regions of the image.

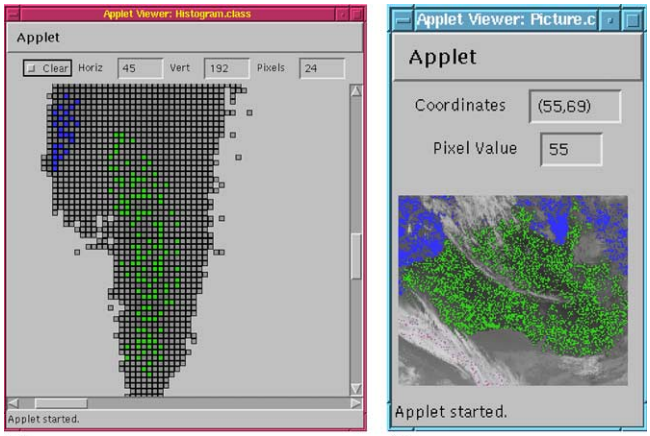

Fig. 4. Java image classifier tool: (a) magnified histogram, (b) image previewer with designed classification highlighted.

If it is found that at the starting stages of a cyclone, there is a certain correlation graph, then image sets may be automatically scanned to contain that graph. This may be useful information for automated forecasting.

Fig. 5 shows some of the interesting features from a simple land/sea/cloud classification using only two channels—visual reflected signal and thermal infrared emitted signal. In this case a very simple scheme has been employed to identify regions in the correlation plot of the two channels, grouping simple polygons of pixel values as "features" and colouring them either brown/green for land; blue for sea or white/grey for clouds. A specular reflection effect in the visual channel arising from an area of sea shows up as an anomalous pseudo-land feature in the classified colour-composite image.

Our package acts as a decision support tool allowing a user to design a classification experiment before launching a batch job which can trawl through the archive processing the required images in parallel and accumulating statistics on the complete data set.

3.4. Coordinate transformations and georectification

The imagery in our repository can be transformed from what is essentially raw pixel coordinates to a mapping onto the Earth's surface (in a specified coordinate systems) using the known geometry, orbit and attitude of the satellite. We are constructing a data parallel computational module that performs this transformation, known as georectification, on image data. This is a computationally intensive task, requiring a significant number of trigonometrical calculations to be carried out for each pixel transformed. Two approaches are useful. The satellite image can be rectified pixel by pixel to map it onto longitude and latitude. This requires a series of trigonometrical and matrix operations to be carried out for each pixel. A computa-

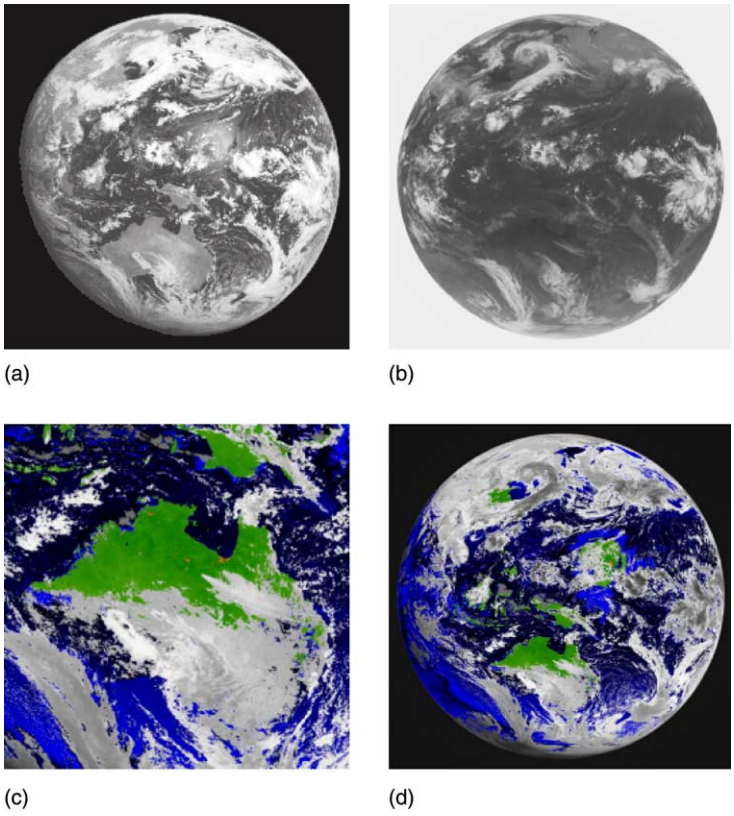

Fig. 5. GMS-5 data: (a) visible spectra, (b) thermal IR1 spectra, (c) thermal IR2 classified Australia and (d) classified world.

tionally cheaper approach is to warp the satellite image to align with a desired map of earth. This involves identifying points on the satellite image that correspond with known points on the map, for example an image classifier could be used to produce a land–sea mask that can be aligned with vector data specifying the known coastline boundaries for a particular geospatial coordinate system. Fig. 6 shows an example of a land–sea mask around Australia generated as seen by the GMS-5 satellite.

4. Parallel spatial data interpolation

GIS applications commonly use a regular 2D or 3D grid, such as pixels in a satellite image, a latitude–longitude coordinate grid on the surface of the earth, or the 3D grid of points modelling the atmosphere and the earth's surface for weather forecasting. In some situations, data required as input for an application, or collected for ground truthing a simulation, may not be co-located with grid coordinates used in

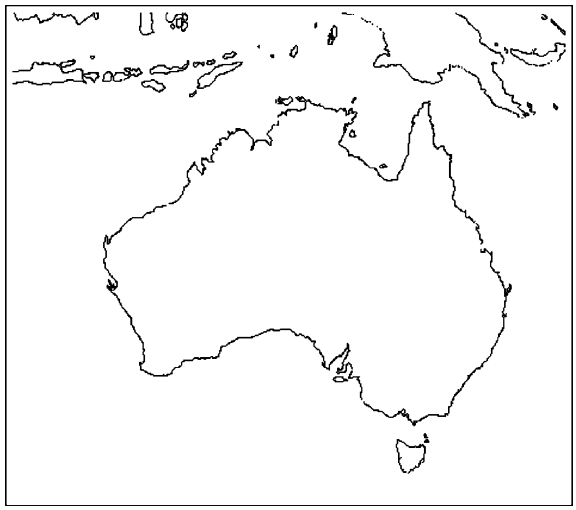

Fig. 6. Land–sea mask over typical region as seen by GMS-5 satellite.

the computation, and may be highly irregular in its spatial distribution. In order to align this data with the grid coordinates, data interpolation is required.

In this section we describe two applications for which complex and computationally intensive spatial data interpolation was required, and discuss parallel implementations of the interpolation algorithms.

In the first case the aim was to develop a specialised parallel program that would be integrated with a parallel weather forecasting program, while in the second case the idea was to provide a general-purpose interpolation program that could be run on a parallel machine and offered as a service to provide faster or more accurate interpolation of spatial data within a distributed GIS framework.

4.1. Data assimilation for weather forecasting

The Unified Model (UM) is a Fortran code employed by the UK Meteorological Office for numerical weather prediction and for climate prediction. The UM allows various resolutions and options so that one suite of code can carry out all the required functions. Broadly speaking, a weather prediction code consists of three main components: the atmospheric dynamics; the physics calculations; and the assimilation of observational data. The dynamics of the UM is based around a regular rectangular grid based model and suggests a number of natural strategies for data decomposition onto a regular grid of processors in a parallel computing system. A number of authors have considered the basic parallel operations necessary for implementing the dynamics of weather models [2,13,25,60,62]. The physics calculations involve localised computations being characterised by nested conditional blocks of code [3].

The assimilation of observational data is by no means trivial and it is still difficult to identify a *natural* way to implement this on the massively parallel systems available now and in the foreseeable future. A common approach that has emerged from multi-processor capabilities in the mid to late 1990s is to adopt an ensemble or "Monte Carlo" approach to weather models and associated data assimilation. In such an approach several forecast simulations are launched with different but similar input parameters. A convergence or agreement amongst different runs of the model is interpreted as a high probability of correctness of the forecast. Cluster computers or clustered supercomputers are usefully applicable to this ensemble approach. The data assimilation scheme is vital to the success of a numerical weather prediction (NWP) code and a one day assimilation can cost up to twice as much in elapsed execution time as a single day forecast. This ratio is likely to increase in future given the expected trend in observation volumes [3].

It emerged from a feasibility study of the whole UM code [25] that, in consequence, the data assimilation scheme was a suitable target for serious investigation and possible implementations on parallel systems. If this component of NWP codes cannot be efficiently implemented on parallel systems, it will severely limit their utility compared to the vector machines traditionally used for meteorological applications.

As indicated above, the dynamics component of the UM suggests that a regular data decomposition strategy will be most natural for a parallel implementation on a distributed memory parallel system [25]. Our work focussed around exploring the means of migrating the data assimilation component of the UM onto a data parallel language such as High Performance Fortran [32]. A data parallel language approach was originally chosen for ease of software development although an MPI [46] message passing approach was later investigated (using the same decomposition algorithms) to improve efficiency.

The data assimilation scheme employed in the UM model is known as the Analysis Correction (AC) scheme [41,65]. As formulated at present this is an iterative analysis scheme with each iteration interleaved by a dynamics and a physics step. The assimilation scheme is being continually developed and different algorithms may be employed in future which make a more optimal use of observational data and which may be better suited to future operational observation systems. Nevertheless it is a useful exercise to investigate how the current scheme can be implemented on massively parallel computer systems as the results from this will feed directly into the planning for future systems.

Data assimilation for the UM at the time of this work was done using variational analysis [41]. We will not dwell on the numerical details of this scheme but will concentrate on the computational aspects for an efficient implementation. Furthermore, we restrict attention here to the analysis correction necessary for a single level of the atmosphere and focus on the horizontal analysis of surface pressure data. It is expected that a parallel implementation of the UM would only involve decomposing the model fields horizontally [25,63] so that the serial loops over atmospheric levels would increase the computational load on individual processors in our implementations but would not affect the basic decompositional strategies described. For the

purposes of obtaining quantitative performance figures we also restrict ourselves to the model field for pressure only. This makes our implementation experiments simpler at the cost of further reducing the computational load on individual processors.

The AC scheme essentially involves the interpolation of observed data onto the regular grids used by the UM. The observed data will typically originate from: radiosondes; surface measurements; aircraft reports; satellite cloud vector data; satellite sounding data and satellite scatterometer data. The model fields of temperature, pressure, wind and moisture are influenced by a variety of these sources. The observational data are not at particular points on the model grids but are rather scattered across the surface of the globe. Furthermore, the area of influence of a particular observation is not trivial. The constraints of the AC scheme require that these areas of influence or "footprints" vary considerably in size and distribution. A vector of observations can therefore be assembled at each time step of the UM, but these vectors vary in length considerably and are not presented to the UM in a simple sorted order.

Fig. 7 illustrates the wide variation in the number of observational influences per model grid point. The continents can be readily discerned from this data and the highest peak is around Reading and Bracknell. The figure shows observation data points or events as they are allocated to a geometric decomposition of atmospheric data in a weather model. In general there is a considerable load imbalance among processors according to this scheme. Fig. 7 shows the magnitude of the problem where the population histogram of observations over area is shown. The peak is around Bracknell in the UK, where the UK Meteorological Office has most obser-

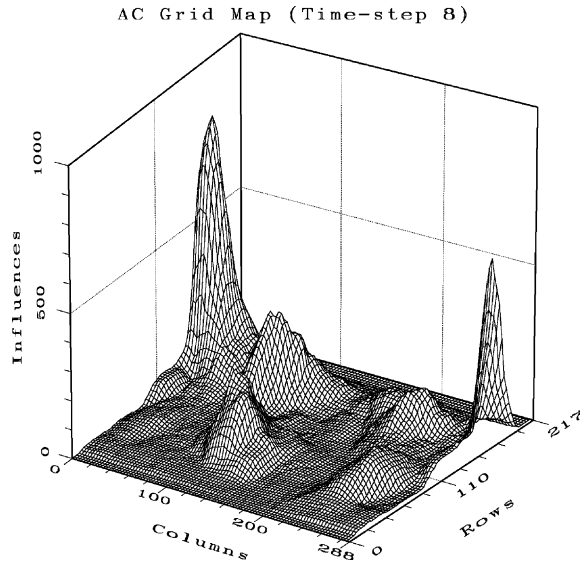

Fig. 7. Inhomogeneity of observation influences per grid point in space. The major peak corresponds to the location of the UK Meteorological Office in Bracknell, England.

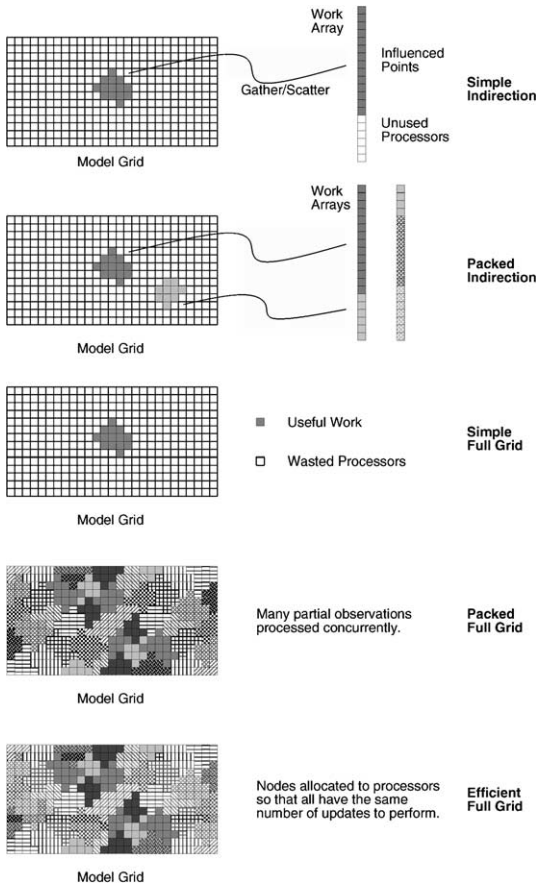

Fig. 8. Strategies for load balancing.

vational data available [25]. This load imbalance is changing as weather forecast organisations use more and more satellite data to be assimilated into operational models.

One way of approaching this problem is to find a mapping that shuffles or folds the geometrically allocated data so processors receive on average the same number of observational data processing tasks (see Fig. 8). On architectures where data shuffles are relatively fast (cheap in resources) this is a good approach. Architectures which have a fast bus or other communications infrastructure to interconnect processors meet this criterion.

4.2. Kriging interpolation for rainfall analysis and prediction

Fusion of large remotely sensed data sets with directly observed data for ground truthing is an important and computationally demanding problem. Satellite imagery

data generally has a regular grid pattern completely different from the distribution of ground truth data. The data fusion requires interpolation of the irregularly spaced ground truth data with the regular grid of the satellite image pixels.

A typical instance of this problem arose in a project to investigate correlations between vegetative growth cover and rainfall conditions at high ground resolution. Data from the GMS-5 satellite, particularly the infrared channel sensitive to water vapour (see Fig. 2(d)), can be used in a classification system [6,31] to provide an accurate identification of rain-bearing clouds and their movements. Ground truthing data is available in the form of the "Rainman" data set, representing Australian Bureau of Meteorology ground station measurements of rainfall for the whole of Australia. Fig. 9 shows the location of Rainman collection sites around Australia. This data can be combined with vegetation index data derived from the visible spectrum channels of other satellites such as the NOAA series, to provide correlation data between rain and rain-bearing clouds and subsequent changes in vegetation cover [5].

Rainfall analysis and prediction is an example of a GIS application that requires access to multiple large-scale distributed data sets and computationally intensive modelling, so the use of high-performance parallel and distributed computation is desirable.

Given a set of known data values for certain points we can interpolate from these points to estimate values for other points, for example on a regularly spaced grid. Linear polynomial techniques provide a method for sampling arbitrary points from gridded data, but the problem is more complex if the source data is sampled at arbitrary points. Schemes such as the variational analysis for data assimilation used in numerical weather forecasting are difficult to apply for very large data sets. One technique for tackling this problem is known as Kriging interpolation [40,42,51], which uses a weighted sum of the known points to estimate the value at an unknown point. This technique can be formulated as a matrix problem connecting the data sets, and the weightings computed using standard matrix solver techniques such as LU facto-

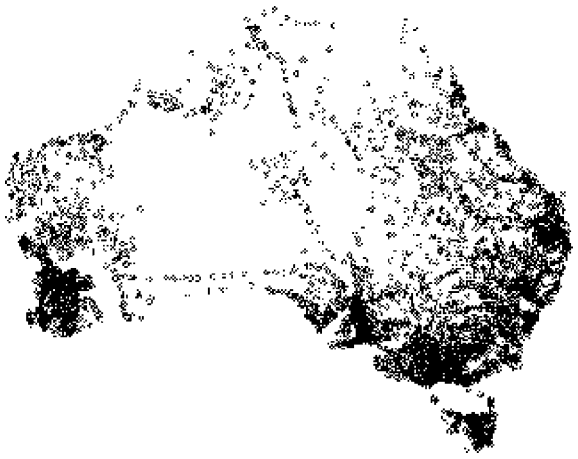

Fig. 9. Location points for Rainman data collection.

risation. The matrices produced are effectively dense and need to be solved using a full matrix method. This method works best for known values that are not evenly scattered [51]. As can be seen in Fig. 9, the rainfall data that we wish to interpolate is clustered, for example there are more known data values for cities and more populated regions, and is therefore well suited to Kriging interpolation.

Generally the estimation of an unknown point only takes a limited range of the known values into consideration. This is done for two reasons. Firstly, known values at a great distance from the unknown point are unlikely to be of great benefit to the accuracy of the estimate and secondly, the operation is less expensive. However, in general Kriging produces the best results when the largest possible number of known points is used in estimating each unknown point [4]. Obviously this is the most expensive option and hence it is desirable to develop a means of implementing the Kriging algorithm on a high-performance parallel computer.

The Kriging problem is an interesting example of a computationally intensive processing component in a GIS. Many of the spatial data manipulation and visualisation operations in a GIS may be relatively lightweight and capable of running on a PC or a low performance computing platform, but Kriging a large data set is too computationally demanding and therefore needs to be run on a separate "accelerator platform" such as a parallel computer, accessed remotely via a distributed computing framework.

We implemented the Kriging algorithm on a large dedicated parallel computer (a 128 processor Connection Machine CM5) as well as a network of workstations (12 Alpha workstations connected by a high-speed ATM network) [40]. These represent the two major classes of computational resources we envisage will be available in a computational grid. We believe this work was the first parallel implementation of the Kriging algorithm. The program was developed using data parallel Fortran—Fortran 90 was used on a single workstation, High Performance Fortran (HPF) on the cluster of workstations, and CMFortran on the CM5. On the CM5 we employed the LU factorisation utility in the CM Scientific Software Library (CMSSL) for solving the matrix component of the Kriging algorithm. On other parallel platforms the ScaLAPACK library [8] could be used.

Fig. 10 shows the interpolated surface produced using Kriging for a selection of Rainman data for South Australia.

Performance for the Kriging algorithm depends on both the number of known data points and the size of the target surface. The time for the LU matrix solver is a large proportion of the total compute time and grows as N^3 for an $N \times N$ target surface grid. We found that interpolating 100 points onto a 500×500 target surface grid took on the order of 10 min on a single processor workstation. The Rainman data contains up to 4000 known points, and the GMS-5 image of Australia is on the order of 1000×1000 pixels. This problem size is too large to run in the memory available on a standard workstation, and if enough memory were available it would take over an hour to run. Fortunately LU matrix solvers can be efficiently parallelised on most parallel architectures (this is the basis for the well-known Linpack benchmark used to rank supercomputer performance), so the time for Kriging interpolation can be greatly reduced by using a parallel machine. For more detailed performance results, see reference [40].

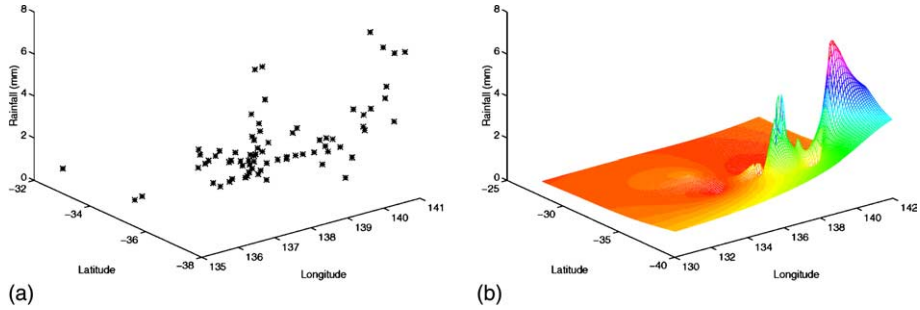

Fig. 10. Interpolated rainfall surface for a selection of known rainfall data: (a) known data points; (b) interpolated surface using Kriging.

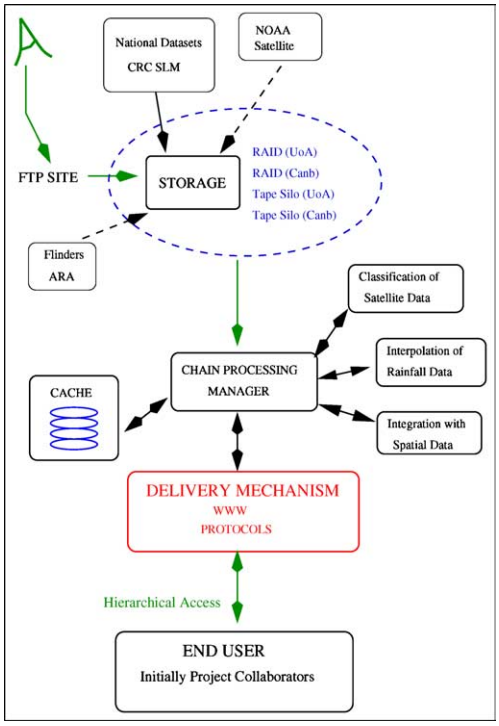

Fig. 11. Distributed computing framework for rainfall prediction.

The algorithms described are still relevant for modern cluster computing systems and for some of the emerging new processor architectures that support data parallelism at a chip level or which employ ''assist processors'' or threads to support fine grained data parallelism.

The measurement and prediction of rainfall and the analysis of its effects on vegetation requires ready access to long sequences of satellite images in order to carry out

the time correlation between rainfall and vegetation. We have developed prototype online storage systems for handling efficient remote access to data by such applications. The Kriging interpolation service needs to be integrated into this distributed computing framework in order to provide a complete system. Fig. 11 shows the processes involved in this application:

- collection of the satellite and Rainman data,
- user request and interpretation of the request,
- execution of the application on the selected data,
- return of the application results to the user.

Fig. 11 also shows the specification of storage equipment available and the access paths for user communication and data access. The Chain Processing Manager shown in the diagram performs the interpretation of the user request and handles scheduling of execution upon the selected resource. Caching is also shown connected to the Chain Processing Stage. This caching involves caching intermediate results and final results from operations to increase efficiency for repeated requests.

5. Distributed workflow-based defence applications

In this section we describe a spatial data processing application prototype we developed with the Australian Defence Science and Technology Organisation (DSTO) to demonstrate a distributed computing framework embedding specialist processing components. The application involves the detection of small targets using a set of heuristics and algorithmic components used in the defence community. Our goal was to implement a workflow system that would allow processing of archived or remote imagery using a sequence of processing commands designed interactively by a user.

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 [49] to address this problem, however it is currently much less mature than the GIAS specification, and a lot more work still needs to be done to create a comprehensive framework that is efficient and easy to use. Sun are also tackling the issue of image processing workflow with their Java Advanced Imaging (JAI) [61] package. We are working with DSTO on an implementation of the GIXS using JAI, and investigating potential improvements to both of these specifications.

For the purposes of discussion we consider the reconnaissance GIXS use case [9,67] shown in Fig. 12. This use case may be typical of an image analyst user's processing request. The steps we consider are:

- request an image from an imagery archive,
- submit the image to a small target detection algorithm,

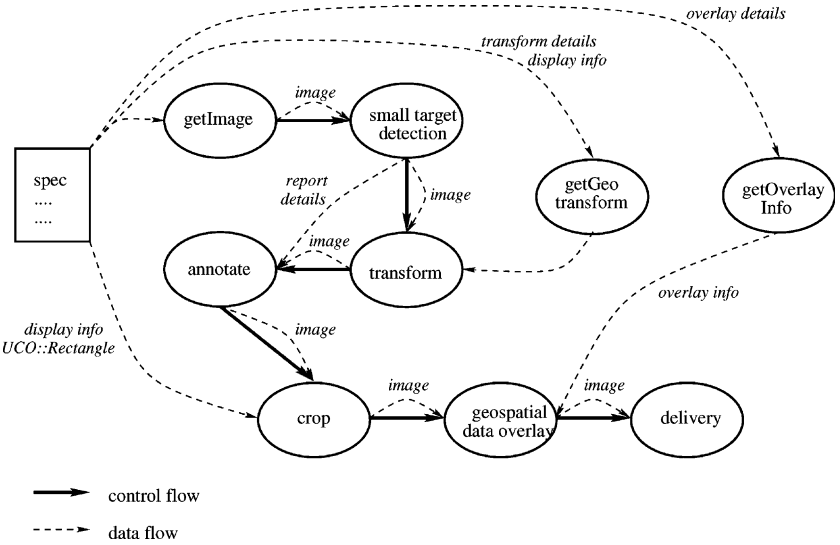

Fig. 12. An image exploitation use case. The specification on the left represents the user's processing request. The specification details image exploitation services (IESs), represented by boxes, which consume and produce data in a processing chain. Thick arrows represent control during execution of the processing request. Thin splines represent the exchange of data that occurs between IESs. This use case details what may be a typical image analyst's processing request: request an image from an image archive, perform some small target detection on the image and annotate the image with the results of the target detection algorithm, transform and crop the image to a suitable dimension and resolution for display and overlay any available geospatial data before presenting to a human for verification.

- transform the image for displaying (i.e. aligning North, etc.),
- annotate the image with the results of the detection algorithm,
- crop the image for displaying,
- overlay some other relevant geospatial data for the region,
- display the image for human verification of the detection process.

We have developed a prototype system, implemented using Java, JAI and CORBA, that (mostly) conforms to the GIXS and handles this use case. We have had to modify or extend some of the GIXS interfaces to deal with the practicalities of a federated system where processing and data access can be distributed [9].

Within the GIXS [49] model, the user composes a request for image exploitation services (IESs) that they wish to have executed on their behalf. As Fig. 12 shows, even a simple application as we describe here can rapidly become quite complex in terms of dataflow and data management. The actual representation of such a processing chain is not clearly specified in the GIXS. The problem of processing-chain representation is compounded when a federated model of computation is considered [68], whereby different processes may be executed on different servers. Amongst others the following issues must be addressed:

- How are results specified by the user?
- How are multiple results specified in a call-back situation?
- Where should the IESs be sourced from?
- How should processing chains be split across nodes in the federated system?

We consider these issues and possible approaches, drawing from discussion of the DISCWorld [29,34] metacomputing environment. One of the fundamental differences between the GIXS and DISCWorld models is that *all* results (and intermediate results) in DISCWorld are significant and possibly valuable to the client and the remainder of servers in the system. In the GIXS, typically only the final, *sink* node in the user's processing request is required. Both the GIXS and DISCWorld accept a user request in the form of a directed acyclic graph (DAG), with the nodes specifying the services to be executed and the edges representing data inputs and outputs.

We suggest the annotation of the DAG nodes in such a manner as to inform the system whether the results of the nodes' computations are of interest. Thus, if clients are interested in more than the final output of the DAG, they should have the option of having a reference to the results returned to them. In addition, when the GIXS framework decomposes a user's DAG into sub-DAGs for execution on different hosts, a facility should be made available for registration of interest by a host on the result of a sub-DAGs outputs.

There are a number of different ways by which the user's processing request (DAG) can be represented. In the DISCWorld prototype DAGs are represented by use of the Distributed Job Placement Language (DJPL). The DJPL is an

```
<?xml version="1.0"?>
<!DOCTYPE DJPL SYSTEM "djpl.dtd">
<DJPL> <USER> <ID>heath</ID> <GROUP>dgis</GROUP>
            <PERMISSION>unrestricted</PERMISSION> </USER>
       <JOB> <ID>00005</ID> <PRIORITY>5</PRIORITY> <COST> <SOFT>100</SOFT>
             <MAX>120</MAX> <ESTIMATE>90</ESTIMATE> </COST> <TIME>360</TIME>
             <SERVER><NAME>opal.cs.adelaide.edu.au</NAME><PORT>6668</PORT>
             </SERVER> </JOB>
  <ALIASES>
  </ALIASES>
  <INSTRUCTIONS>
    <INSTRUCTION> <SERVICE>findImage</SERVICE>
          <PARAMETER TYPE="INPUT"> <NAME>date</NAME>
             <VALUE></VALUE> </PARAMETER>
          <PARAMETER TYPE="INPUT"> <NAME>time</NAME>
             <VALUE></VALUE> </PARAMETER>
          <PARAMETER TYPE="INPUT"> <NAME>area</NAME>
             <VALUE></VALUE> </PARAMETER>
          <PARAMETER TYPE="OUTPUT"> <NAME>output_image</NAME>
             <VALUE></VALUE> </PARAMETER>
     </INSTRUCTION>
  </INSTRUCTIONS>
</DJPL>
```

Fig. 13. Reconnaissance use case of Fig. 12 expressed in DISCWorld's Distributed Job Placement Language (DJPL). The DISCWorld model assumes that services (IESs) have a global name which is the same, and *known* across the whole system, and that input and output data of services is able to be canonically named.

XML-based language specification which has the benefit of being able to be parsed by any platform that has an XML parser.

An example of the reconnaissance use case (Fig. 12) expressed using the DJPL is shown in Fig. 13.

6. Spatial data grids

6.1. Data grids

Many computational grid projects require remote access to large amounts of data, which has led to the term "data grid" [1] being used to describe grid environments and services that are developed to support such applications. Many scientific data repositories are so large that it is not feasible for all the researchers that use them to have local copies of the data. For example, the latest earth observation satellites produce Terabytes of data per day [47]. Such large data sets will be replicated at only a few sites around the world.

With these very large data sets becoming more common, the communities interested in them must overcome difficulties of locating and accessing the data they need. Huge data sets present new problems for users in identifying the particular data they need. Conventionally, a user maintains a file system and will access data by file name, but it is difficult for the user to know the physical location of all relevant files in a large evolving distributed repository. Hence, it is preferable to reference the required data by its properties, such as GMS-5 satellite image for Adelaide at 2330UTC 26th January 2001. Metadata associated with a data set can be searched to facilitate such content-based access. Thus, data access is easier for the user, but the underlying system must solve problems of where to locate the data, and transmit it to where it is needed. This may necessitate construction of a multi-site database, where the most commonly used data is replicated at multiple sites.

Currently, grid tools to support large-scale data sets are fairly low-level. Globus provides basic file manipulation routines through the Globus Access to Secondary Storage (GASS), and secure high-speed file transfer with GridFTP. The most recent version of Globus also provides a Replica Catalog service for discovery and querying of distributed replicated data archives. We are aiming to provide higher-level data grid services, particularly targeted at geospatial data.

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 [28]; developing intelligent data caching and process scheduling strategies [34]; and designing an efficient middleware infrastructure for supporting distributed active data archives [27,54].

6.2. Virtual data grids

Standard data repositories 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 distributed "active" data archives [26,27,54], also known as "virtual data grids" [12], are particularly useful for scientific and decision support applications, and can greatly enhance the value of static collections of large data sets. By exploiting such services, data that may be on a remote server, in a hierarchical file system (e.g. tape silo), in a database, or may not even exist yet (i.e. "virtual data" that is generated by processing of raw data), can be accessed by the application as though it were a local file. This approach can also improve the efficiency of transporting large amounts of data over a network from a remote data resource, for example by using pre-processing to crop out a region of interest in a large satellite image, or by intelligent pre-fetching and caching of data.

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.

The computational services provided by active data archives must have well-defined interfaces so they can be accessed by remote clients. The client application does not need to be aware of the implementation of the computational service, only the interface required to invoke it. For interactive applications or those requiring near real-time results or faster response time, the processing may be transferred to a high-end parallel 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.

Our OLGAS software [10,11] provides a simple example of an active data archive for geospatial image data. For example, the client can request satellite data for a particular geographical region to be delivered in a specified data format (e.g. JPEG,

GeoTIFF, NITF). In order to find the appropriate region, the data needs to be georectified. If this has not already been done, the data server will initiate this computation on the specified image. The region of interest will then be cropped out of the image, and the resulting data will be converted to the appropriate format for delivery. Another possibility is that the data repository may only hold a subset of the complete data archive for a particular satellite. If it does not have the requested image, it could first access it from another archive that does, and then cache a copy of the image in case it is required again later.

In summary, there are four key components to an active data archive: the low-level bulk data storage management system (for both data and metadata); 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 [26,27,54] 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 as a stand-alone application or as an applet within a Web browser environment.

6.3. Standards for spatial data grids

The OpenGIS Consortium (OGC) has made a good start in developing the standards required for implementing an interoperable distributed spatial data infrastructure. However OGC has just begun to address the many complex issues involved in developing federated systems that utilise multiple distributed data access and processing services. Even with well-defined interface standards, integrating spatial data servers and applications that use different distributed computing technologies is a difficult undertaking.

Grid technologies can potentially provide a solution to this problem, by the use of standard grid protocols and services; the Community Grid (CoG) toolkits [22] for interfacing Globus to existing distributed computing technologies such as CORBA, Java and the Web; and the Open Grid Services Architecture (OGSA) [18] currently under development. Grid technologies would then act as a unifying distributed computing middleware to enable a truly interoperable spatial data grid. Globus also provides a number of additional useful features such as single sign-on security and authentication using the Globus Security Infrastructure (GSI); efficient, secure bulk data transfer using the Globus Access to Secondary Storage (GASS) and the GridFTP secure file transfer protocol; and data grid services such as metadata and replica management services [1].

One of the problems we are attempting to address is to identify how standards for spatial data infrastructure can evolve in a way that is compatible with general grid computing standards. We are aiming to develop a prototype spatial data grid that utilises standard grid tools and technologies and conforms to OpenGIS standards. Many of the issues still being tackled by the OpenGIS and NIMA initiatives are to do with how distributed processing of spatial data can be specified and managed. Grid technologies are clearly applicable here. Ideally the development of interoper-

ability standards to enable distributed data access and processing for a specific application area (such as spatial information systems) should be compatible with the more general standards being developed for grid computing, however at the moment such standardisation efforts appear to be evolving independently, perhaps because grid computing is still an immature technology. One of the goals of our work is to identify how standards for a distributed spatial data infrastructure could evolve in a way that is compatible with general data grid standards.

Recent initiatives such as the Earth System Grid [15], the European Data Grid project on Earth Observation Science Applications [16], and the European Space Agency's SpaceGrid [17] are also developing spatial data grids, and should act as drivers for the development of spatial data grid infrastructure and standards.

7. Summary and future issues

In this paper we have summarised projects that have spanned nearly a decade. During this time we have seen a transition in the maturing and deployment of parallel computing. In the early 1990s massively parallel computing was still a relatively arcane branch of computing. It gained acceptance into the 1990s and by the end of the 20th century we might justifiably claim that massive parallelism had been accepted into the mainstream of computing. Certainly there are now plenty of tools and approaches that make parallel computing accessible to many applications programmers, including those in the field of geospatial and geographic data manipulation. In the period from the mid 1990s to the present metacomputing and grid systems have emerged as a new research area and in 2002 grid computing is showing all the excitement for the computing research community that parallel computing did in the 1980s. Grid systems and embedded parallel resources and services are increasingly accessible to applications users, and it seems likely that the grid will come to a useful state of maturity over the next few years.

Over this last decade there has been enormous development in the power of individual processors and in the memory and disk capacities available to processing nodes. This has partially been due to technological improvements but also due to the simple economics of the mass market for disk and memory components.

Bandwidth cost, capacity and availability has also grown. However there are still fundamental limits to the latencies involved in accessing remote resources. Approaches such as data replication and workflow "chunking together" of job requests into complex queries to support "action at a distance" seem the only feasible solution to latency limitations. Grid systems will have to adopt such approaches to become truly global in scale.

Parallel computing at the massive parallelism level, supported by message passing or language parallelism, remains an interesting area in which to work particularly for applications using large geospatial data sets. Cluster computer systems remain a cost effective way of exploiting this sort of parallelism in applications. However, we believe it is no longer possible nor indeed desirable to treat parallel computing separately from distributed computing. Many of yesterday's parallel computing issues are today's distributed computing problems.

Acknowledgements

We thank S.J. Del Fabbro, C.J. Patten, K.E. Kerry Falkner, J.F. Hercus, K. Hutchens, K.D. Mason, J.A. Mathew, A.J. Silis and D. Webb who helped implement the application prototypes described in this paper, and K.J. Maciunas, F.A. Vaughan and A.L. Wendelborn for their input in developing the grid computing concepts. Thanks also to K.P. Bryceson for useful discussions on data interpolation and mapping. The Distributed and High Performance Computing Group is a collaborative venture between the University of Wales, Bangor and the University of Adelaide. We acknowledge the support provided by the Australian Defence Science and Technology Organization, Sun Microsystems, and the Research Data Networks and Advanced Computational Systems Cooperative Research Centres (CRC) which were established under the Australian Government's CRC Program.

References

[1] W. Allcock et al., The data grid: towards an architecture for the distributed management and analysis of large scientific datasets, Journal of Network and Computer Applications 23 (2001) 187–200.
[2] S.R.M. Barres, T. Kauranne, Spectral and multigrid spherical Helmholtz equation solvers on distributed memory parallel computers, in: Proc. Fourth Workshop on Use of Parallel Processors in Meteorology, ECMWF, 1990.
[3] R.S. Bell, A. Dickinson, The Meteorological Office Unified Model for data assimilation, climate modelling and NWP and its implementation on a CRAY Y-MP, in: Proc. Fourth Workshop on Use of Parallel Processors in Meteorology, ECMWF, 1990.
[4] S. Border, The use of indicator Kriging as a biased estimator to discriminate between ore and waste, Applications of Computers in the Mineral Industry, University of Wollongong, N.S.W., October 1993.
[5] K. Bryceson, M. Bryant, The GIS/rainfall connection, in GIS User, No. 4, August 1993, pp. 32–35.
[6] K.P. Bryceson, P. Wilson, M. Bryant, Daily rainfall estimation from satellite data using rules-based classification techniques and a GIS, unpublished, November 1995.
[7] H. Casanova, J. Dongarra, NetSolve: a network server for solving computational science problems, in: Proc. Supercomputing '96.
[8] J. Choi, J.J. Dongarra, R. Pozo, D.W. Walker, ScaLAPACK: a scalable linear algebra library for distributed memory concurrent computers, in: Proc. of the Fourth Symposium on the Frontiers of Massively Parallel Computation, IEEE Computer Society Press, 1992, pp. 120–127.
[9] P.D. Coddington, G. Hamlyn, K.A. Hawick, H.A. James, J. Hercus, D. Uksi, D. Weber, A software infrastructure for federated geospatial image exploitation services, DHPC Technical Report DHPC-092, Department of Computer Science, University of Adelaide, 2000.
[10] P.D. Coddington, K.A. Hawick, K.E. Kerry, J.A. Mathew, A.J. Silis, D.L. Webb, P.J. Whitbread, C.G. Irving, M.W. Grigg, R. Jana, K. Tang, 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.
[11] P.D. Coddington et al., Interfacing to on-line geospatial imagery archives, in: Proc. of Australasian Urban and Regional Information Systems Assoc. Conf. (AURISA'99), Leura, NSW, 1999.
[12] E. Deelman et al., A virtual data grid for LIGO, Lecture Notes in Computer Science 2110 (2001) 3–12.
[13] D. Dent, The ECMWF model on the CRAY Y-MP8, in: Proc. Fourth Workshop on Use of Parallel Processors in Meteorology, ECMWF, 1990.

[14] Distributed and High-Performance Computing Group, On-Line Data Archives (OLDA) Project. Available from <http://www.dhpc.adelaide.edu.au/olda/>.
[15] Earth System Grid. Available from <http://www.earthsystemgrid.org/>.
[16] European Data Grid, Earth Observation Science Applications. Available from <http://styx.esrin.esa.it/grid/>.
[17] European Space Agency, SpaceGrid. Available from <http://www.spacegrid.org/>.
[18] I. Foster, D. Gannon (Eds.), The Open Grid Services Architecture Platform. Available from <http://www.ggf.org/ogsa-wg>.
[19] I. Foster, C. Kesselman (Eds.), The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, Inc., 1999, ISBN 1-55860-475-8.
[20] G.C. Fox, R.D. Williams, P.C. Messina, Parallel Computing Works!, Morgan Kaufmann Publishers, Inc., 1994, ISBN 1-55860-253-4.
[21] The Globus Project. Available from <http://www.globus.org/>.
[22] The Globus Project, Commodity Grid Kits. Available from <http://www.globus.org/cog/>.
[23] The GMS User's Guide, Meteorological Satellite Center, 3-235 Nakakiyoto, Kiyose, Tokyo 204, Japan, second ed., 1989.
[24] M. Grigg, P. Whitbread, C. Irving, A. Lui, R. Jana, Component based architecture for a distributed library system, in: Proc. Distributed Multimedia Systems Conference, Aizu, Japan, 1999.
[25] K.A. Hawick, R. Stuart Bell, A. Dickinson, P.D. Surry, B.J.N. Wylie, Parallelisation of the unified weather and climate model data assimilation scheme, in: Proc. Workshop of Fifth ECMWF Workshop on Use of Parallel Processors in Meteorology, European Centre for Medium Range Weather Forecasting, Reading November 1992 (invited paper).
[26] K.A. Hawick, P.D. Coddington, Interfacing to distributed active data archives, Future Generation Computer Systems 16 (1999) 73–89.
[27] K.A. Hawick, P.D. Coddington, H.A. James, C.J. Patten, On-line data archives, in: Proc. Hawai'i Int. Conf. on System Sciences (HICSS-34), Maui, 2001.
[28] K.A. Hawick, H.A. James, K.J. Maciunas, F.A. Vaughan, A.L. Wendelborn, M. Buchhorn, M. Rezny, S.R. Taylor, M.D. Wilson, Geographic information systems applications on an ATM-based distributed high performance computing system, in: Proc. High-Performance Computing and Networks (HPCN) Europe '97, Vienna, 1997.
[29] 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, F.A. Vaughan, DISCWorld: an environment for service-based metacomputing, Future Generation Computing Systems (FGCS) 15 (1999) 623–635.
[30] K.A. Hawick, H.A. James, A distributed job placement language, DHPC Technical Report DHPC-070, Department of Computer Science, University of Adelaide, May 1999.
[31] K.A. Hawick, H.A. James, Distributed high-performance computation for remote sensing, in: Proc. Supercomputing '97, San Jose, November 1997.
[32] High Performance Fortran Forum (HPFF), High Performance Fortran Language Specification, Scientific Programming, vol. 2, no. 1, July 1993.
[33] J. Hildebrandt, M. Grigg, S. Bytheway, R. Hollemby, S. James, Dynamic C2 application of imagery and GIS information using backend repositories, in: Proc. Command & Control Research & Technology Symposium, Rhode Island, June 1999.
[34] H.A. James, Scheduling in metacomputing systems, PhD Thesis, Department of Computer Science, University of Adelaide, 1999.
[35] H.A. James, K.A. Hawick, Data futures in DISCWorld, in: Proc. High Performance Computing and Networking (HPCN Europe 2000), Amsterdam, May 2000, LNCS 1823, pp. 41–50.
[36] H.A. James, K.A. Hawick, Eric: a user and applications interface to a distributed satellite data repository, DHPC Technical Report DHPC-008, Department of Computer Science, University of Adelaide, 1997.
[37] H.A. James, K.A. Hawick, A Web-based interface for on-demand processing of satellite imagery archives, in: Proc. of the Australian Computer Science Conference (ACSC'98), Perth, February 1998.
[38] H.A. James, K.A. Hawick, Remote application scheduling on metacomputing systems, in: Proc. HPDC'99, Redondo Beach, California, August 1999.

[39] H.A. James, C.J. Patten, K.A. Hawick, Stencil methods on distributed high performance computers, DHPC Technical Report DHPC-010, Department of Computer Science, University of Adelaide, 1997.
[40] K.E. Kerry, K.A. Hawick, Spatial interpolation on distributed, high-performance computers, in: Proc. of High-Performance Computing and Networks (HPCN) Europe '98, Amsterdam, April 1998.
[41] A.C. Lorenc, R.S. Bell, B. Macpherson, The Meteorological Office analysis correction data assimilation scheme, Quarterly Journal of the Royal Meteorological Society 117 (1991) 59–89.
[42] D.C. Mason, M. O'Conaill, I. McKendrick, Variable resolution block Kriging using a hierarchical spatial data structure, International Journal of Geographical Information Systems 8 (5) (1994) 429–449.
[43] J.A. Mathew, P.D. Coddington, K.A. Hawick, Analysis and development of Java Grande benchmarks, in: Proc. of the ACM 1999 Java Grande Conference, San Francisco, June 1999.
[44] J.A. Mathew, H.A. James, K.A. Hawick, Development routes for message passing parallelism in Java, in: Proc. ACM 2000 Java Grande Conference, San Francisco, June 2000, pp. 54–61.
[45] J.A. Mathew, A.J. Silis, K.A. Hawick, Inter server transport of Java byte code in a metacomputing environment, in: Proc. of Technology of Object-Oriented Languages and Systems Pacific (TOOLS 28), Melbourne, November 1998.
[46] Message Passing Interface Forum, MPI: a message passing interface standard. Available from <http://www.mpi-forum.org/>.
[47] NASA Goddard Space Flight Center. Available from <http://www.gsfc.nasa.gov/>.
[48] U.S. National Imagery and Mapping Agency (NIMA). Available from <http://www.nima.mil/>.
[49] U.S. National Imagery and Mapping Agency (NIMA), Geospatial and Imagery eXploitation Services (GIXS) specification, S1010420-B, version 1.2, June 2000. Available from <http://www.nima.mil/sandi/arch/addinfo.html>.
[50] U.S. National Imagery and Mapping Association, USIGS Geospatial and Imagery Access Services (GIAS) specification, version 3.1, N0101-B, February 1998.
[51] M.A. Oliver, R. Webster, Kriging: a method of interpolation for geographical information systems, International Journal of Geographical Information Systems 4 (3) (1990) 313–332.
[52] OpenGIS Consortium. Available from <http://www.opengis.org>.
[53] Open GIS Consortium, Catalog Interface Implementation Specification (version 1.0). Available from <http://www.opengis.org/techno/implementation.htm>.
[54] C.J. Patten, K.A. Hawick, Flexible high-performance access to distributed storage resources, in: Proc. 9th IEEE Int. Sym. on High Performance Distributed Computing (HPDC'00), Pittsburgh, 2000.
[55] C.J. Patten, F.A. Vaughan, K.A. Hawick, A.L. Brown, DWorFS: file system support for legacy applications in DISCWorld, in: Proc. 5th IDEA Workshop, Fremantle, February 1998.
[56] M. Rezny, Aspects of high performance computing, Ph.D. Thesis, Department of Mathematics, University of Queensland, 1995.
[57] A.J. Silis, K.A. Hawick, World Wide Web server technology and interfaces for distributed, high-performance computing systems, DHPC Technical Report DHPC-017, Department of Computer Science, University of Adelaide, 1997.
[58] D. Skillicorn, The case for datacentric grids, in: Proc. Workshop on Massively Parallel Processing, Int. Parallel and Distributed Processing Symposium (IPDPS), Fort Lauderdale, April 2002.
[59] L. Smarr, C.E. Catlett, Metacomputing, Communications of the ACM 35 (6) (1992) 44–52.
[60] M.J. Suarez, Atmospheric modelling on a SIMD computer, in: G.-R. Hoffmann, D.F. Snelling (Eds.), Multiprocessing in Meteorological Models, Springer-Verlag, 1988.
[61] Sun Microsystems, Java advanced imaging API. Available from <http://java.sun.com/products/java-media/jai/>.
[62] S.F.B. Tett, A massively parallel algorithm for the spectral method, in: Proc. Fourth Workshop on Use of parallel processors in meteorology, ECMWF, 1990.
[63] S.F.B. Tett, Ph.D. Thesis, University of Edinburgh, 1992.
[64] J.A. Toon, Intelligent systems: practical data mining in a world saturated with data, DHPC Technical Report DHPC-017, School of Informatics, University of Wales, Bangor, July 2002.
[65] U.K. Meteorological Office, Unified Model Documentation paper 50, 1991.

[66] D.L. Webb, A.L. Wendelborn, K.A. Maciunas, Process networks as a high-level notation for metacomputing, in: Proc. Int. Parallel Programming Symposium (IPPS'99), Puerto Rico, 1999.
[67] D.C. Weber, Reconnaissance use case with GIXS, DSTO note, unpublished, 2000.
[68] D. Weber, H. James, A federated GIXS model, in: Proc 5th Int. Conf. Command and Control, Research and Technology Symposium, Canberra, Australia, October 2000.