

Communication over a Secured Heterogeneous Grid with the GriddLeS runtime environment

Jagan Kommineni, David Abramson and Jefferson Tan
Monash e-Science and Grid Engineering Lab
Faculty of Information Technology
Monash University, Caulfield 3145 Victoria, Australia
{jagan,davida,jtan}@csse.monash.edu.au

Abstract

Scientific workflows are a powerful programming technique for specifying complex computations using a number of otherwise independent components. When used in a Grid environment, it is possible to build powerful “virtual applications” across multiple distributed and heterogeneous resources. Whilst toolkits such as Globus virtualize many system attributes, and thus make it easier to span different organizations, inconsistent security policies and resource heterogeneity can limit the applicability of workflow techniques.

In earlier work, we have described a novel run time environment, GriddLeS, that supports flexible communication patterns between workflow components. GriddLeS abstracts IO operations, such that applications are given the illusion of operating on a local file system, whilst in fact they send and receive data between components. In this paper, we describe how GriddLeS assists in resolving some of the issues that arise due to heterogeneity in security policies and system architectures in a Grid environment. We illustrate the solution using a real world scientific workflow for climate modeling, and demonstrate a system that spans multiple conflicting security domains with heterogeneous resources.

1. Introduction

Computational and Data Grids [1][2][3][4] have been proposed as the next generation computing platform for solving large scale problems in science, engineering, and commerce. They couple geographically distributed resources such as high performance computers, clusters, workstations, and scientific instruments. In the last decade there has been a substantial increase in conducting experiments on geographically distributed resources unified to act as a single *virtual computer*. Early efforts in Grid computing concerned linking supercomputing facilities [1], but recent efforts have gone much further. Scientific applications [5][6][7][8] can be combined as

a collection of interacting software components executing on heterogeneous resources connected by a network. These so called *virtual applications*, or *workflows* [9][10][11][12], are composed of relatively large-grain computations that are assembled into a higher-level program.

Workflows make it possible to process data from an arbitrary source, including databases, real time instruments and other applications. Workflows give the appearance of a single application while actually running computations across different physical organizations and resources. When workflows are built from legacy application components, data is usually moved between them by copying files from one system to another. This is because most legacy applications assume that their data is held in a local file system. As a result, these workflows must be executed synchronously, because a downstream application cannot start until all of its input files are available. This in turn precludes opportunities for overlapping source and consumer components, and restricts the opportunities for optimizing the workflow performance.

Recently, we have built a library called GriddLeS [13][14][15][16] that provides a powerful and flexible communication mechanism between legacy components of a workflow. GriddLeS provides a wide range of mechanisms for building workflows without the need to fix the IO mechanism at design time. It does this by overloading conventional file IO operations to support either local, remote or replicated file access, or direct communication over pipes. When the program is coded to read or write to a local file, GriddLeS may choose an alternative transport at runtime. The choice of transport is made late in the workflow configuration process, providing significant flexibility in building workflows from pre-existing standalone applications.

Previously, we have reported success running components in a homogenous Grid environment [13][14]. On the other hand, when dealing with *heterogeneous* resources, workflows must deal with a

different set of problems arising from incompatible machine architectures and communication capabilities. A specific class of problems arising from architectural conflict is characterized by workflow components that use conflicting data formats. We refer to this as a *data heterogeneity* problem. A separate class of problems arising from conflicting organizational security policies is characterized by workflow components that use different modes of communication. We refer to this as a *security heterogeneity* problem.

Data heterogeneity problems cause data format incompatibilities arising from unique architectural constraints that are sometimes forced upon applications by the resources hosting them. Such problems have been addressed in the past using abstractions between the application and the underlying file system [7]. NetCDF (Network Common Data Form) [17] is a good example of a solution that uses an abstraction layer to facilitate transparent data access. It does this by storing data in a manner that is independent of the system architectures, and provides conversion routines between the machine independent representation and the specific architectures. Another form of abstraction is to communicate data between systems using one of the markup languages such as XML (Extensible Markup Language) [18]. Scientific communities, such as the atmospheric sciences community [7][8], have used the netCDF approach extensively, and thus we have adopted it for the work reported in this paper. One limitation in solutions like netCDF [17] is that they only handle I/O operations over local files. However, as discussed earlier, using local files in this way precludes overlap in the components, restricting the options for tuning the workflow performance. To solve this problem, we have hosted netCDF over GriddLeS. This allows netCDF to inherit the flexible I/O capabilities of GriddLeS, enabling workflow components to perform file I/O beyond the local file system.

Security heterogeneity problems are often caused by conflicting firewall restrictions and represent a serious limitation to building real applications on the Grid. Channels and protocols that are allowed in some organizations on their networks may not be allowed in others. However, most organizations allow communication through a small number of authorized ports and protocols. One such commonly open channel is port 80 using the Hypertext Transfer Protocol (HTTP). To exploit such situations, we have enabled GriddLeS to connect workflow components using one or more proxies via HTTP on port 80.

In this paper, we show how GriddLeS can be adapted to solve the data and security heterogeneity problems. A large atmospheric science application is used to illustrate our solutions which were tested on a

Grid test bed that exhibits both kinds of problems. The test bed consists of a 224 vector processor NEC SX6 [19], a 124x R14000A 600MHz MIPS processor shared memory SGI Origin 3000 [20] and a 97 node, 194 CPU processor Linux Xeon Cluster [21]. These systems are located at three different Australian organizations, namely the Bureau of Meteorology / CSIRO High Performance Computing and Communications Centre, VPAC (Victorian Partnership for Advanced Computing) in Melbourne and the Queensland University of Technology in Brisbane. The communication between these systems is limited because of firewall restrictions.

2. Grid Workflows

A workflow takes a number of otherwise independent components and assembles them together to build complex computations. There are numerous workflow tools that have been developed in recent years. One such system, Kepler [9], is an open-source cross-project, cross-institution collaborative workflow tool that targets the scientific community. It is built on top of the Ptolemy II data flow engine [22]. Kepler has been used in a range of projects in multi-disciplines including eco-informatics, bio-informatics, and geoinformatics, to build workflows.

Kepler supports the construction of arbitrarily complex workflows by combining a series of application components. Execution is controlled by *directors*, which are responsible for defining the orchestration semantics of the workflow. Two directors SDF (Synchronous Data Flow) and PN (Process Network) are most commonly used with Kepler workflows. The SDF director executes components synchronously. Thus, downstream components are not started until all of the inputs are available. The PN director, on the other hand, runs all components in parallel. By switching directors, it is possible to change the performance of the workflow. Clearly, the PN has the potential to make the workflow run faster, but this can only occur if components output data continuously. Whilst it is possible to alter the choice of director without changing the workflow itself, SDF and PN actually require applications to be written specifically for them. For example, if a SDF director is used, then files must be copied from one resource to another as the need arises. In this case, the component reads and writes files using conventional file primitives. On the other hand, if a PN director is used, all the application components run concurrently in a pipelined manner and inter-process pipes are required for IO. These two IO methods, copying or using pipes, are normally different enough that the source code of the

components must be altered if one is chosen over the other. However, GriddLeS supports both file IO and inter-process communication, including the use of pipes, by abstracting conventional IO primitives, such that source modification is not required [13][14], and thus makes it possible to switch directors without changing the actor source.

Figure 1 illustrates part of a simple atmospheric science workflow that we have built using Kepler. This workflow can either be controlled by a PN Director, in which all components start their computations at the same time and can run in parallel, or a SDF director, in which case they are executed synchronously. When run in parallel, the atmospheric models produce time-stepped data information as the computation progresses step by step. This information is immediately transferred to a downstream computation which can start execution before the driving one is completed. The data is streamed by a separate thread which makes the execution and data transfer tasks go in parallel. It is also possible to schedule all the components sequentially using a SDF director instead. In this case the components are executed one after the other and the data is transmitted at the end of computation as a single file copy..

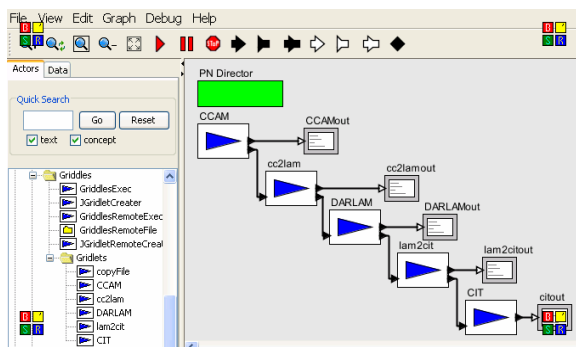


Figure 1. Sample atmospheric science workflow.

3. GriddLeS Architecture and Abstractions

3.1 Griddles Abstraction Mechanism

GriddLeS [13][14][15][16] provides a rich set of interprocess communication mechanisms between application processes. Applications can access a wide variety of resources such as local files, remote files and replicated files in a unified way through standard POSIX IO calls. GriddLeS supports file-based communication between software components by trapping application IO system calls and mapping these

operations dynamically to appropriate services [13][15].

Figure 2 shows an architectural view of GriddLeS. Typically, when a program writes data, the GriddLeS runtime traps the call and can redirect it to a local file, a remote file, a replicated file or connect to a read operation on a local or remote process using a shared buffer. The latter mode allows an overlap in IO operations and model executions. This leads to an efficient solution for stream-based computations.

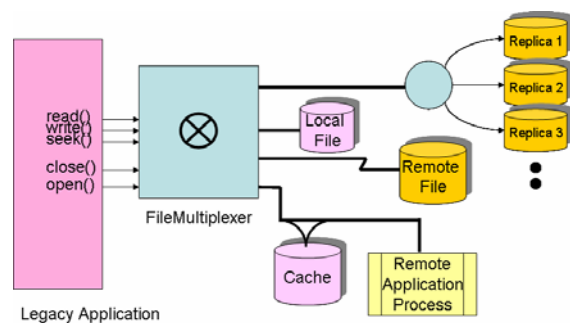


Figure 2. The GriddLeS Architecture.

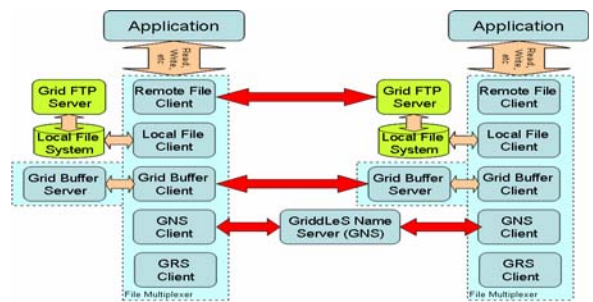


Figure 3. The GriddLeS communication semantics.

The GriddLeS runtime is composed of a number of client modules, specifically a Local File Client, a Remote File Client, a Grid Buffer Client, a GNS (GriddLeS Naming Service) Client and a GRS (GriddLeS Replica Service) Client, as shown in Figure 3 [13][14]. These modules are used to communicate with the appropriate services. The Local File Client is used to access local files. The Remote File Client is used to access remote files using any one of the available file transfer mechanisms, e.g., secure copy (SCP)[23], GridFTP[24], or Storage Resource Broker (SRB) [25]. The GRS [16] client is used to access replicated files and leverages a number of different replication systems including the SRB [25], the Globus Replica Location Service (RLS) [26] and GFarm [27].

The Grid Buffer client is used to communicate across systems using a shared buffer, and facilitates pipe-like communications. When the application tries to open a file, the GriddLeS runtime uses the GNS client for resolving the type of file and its location.

3.2 Implementing netCDF over GriddLeS

NetCDF was designed by climate and earth systems communities for solving data heterogeneity problems that occur when files are transferred from one machine to another. As shown in Figure 4a, netCDF sits between an application and the local file system and performs data conversion for application components. NetCDF is a library module which can be attached to any application component by making modifications to an IO module of the application. Some of the climate models used in this investigation have been modified to use the netCDF module, and thus this makes it an ideal solution to the data heterogeneity problem.

NetCDF does not have capabilities for streaming data over pipes, and thus cannot be used directly with PN types of workflows. However, by coupling netCDF with GriddLeS, as shown in Figure 4b, application components can perform communication between different types of resources because the data is mapped into an architecture-neutral format before it is transferred. Using this solution, we are able to couple applications on different types of resources, and they can communicate directly using the GriddLeS communication mechanisms

3.3 Handling Heterogeneous Security Policies

The Grid resources considered in this paper are located in independent and secured environments with highly restrictive firewall policies. Most of the resources in these environments are composed of a hierarchy of machines – typically front nodes that act as a gateway and execution nodes that perform the work. Execution nodes typically have access only to the front node of the facility, while the front node itself cannot access the outside world directly. Incoming access is also heavily restricted, and the front and execution nodes are usually not accessible from the outside on any port. However, these organizations do not block web (HTTP) traffic through to the public web servers which sit on a demilitarized zone (DMZ) outside the firewall. While the public web server is not accessible from the execution nodes, it usually is from the front node. It is thus possible to create a bridge across firewalls through the public web servers of the two organizations using HTTP connections. We have therefore designed a web-based IO mechanism for GriddLeS that facilitates connections between execution nodes of different

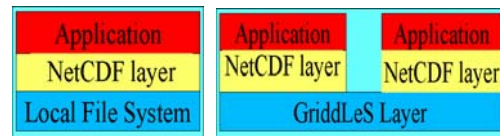


Figure 4a & 4b. Layered non-distributed and distributed environments.

domains using their public web servers and front nodes.

The bridge between domains consists of three segments. First, we establish connections between an execution node and the public web server of the same domain. Second, we establish the connection between the public web server from one domain to that of the other domain. Lastly, at the remote domain, we connect the web server with an execution node. These segments are constructed using lightweight *redirection services* on the front nodes and *buffer services* on the public web servers. The redirection services divert traffic from the execution nodes to the public web server, where the buffer service creates connections across to the other domain’s public web server. The remote buffer service accepts connections and forwards them to the local front node, where the redirection service brings the connection to one of its execution nodes. Out of the two buffer services, the one local to the reader application’s execution node stores the data and the other buffer service acts as a proxy to transfer the data across. By doing this, the reader application component receives data with minimal delay. On the other end, the writer application component transmits data concurrently during its execution.

Figure 5 shows a typical scenario of the data flow pattern when two application components are executing at two independent secure locations. The inter-component communication involves four redirections. The data flow within the organization depends entirely on its own security policies. The assumptions are that the execution nodes have open access to the front node, and public web servers are open to the Internet as well as to the front node of their respective organizations.

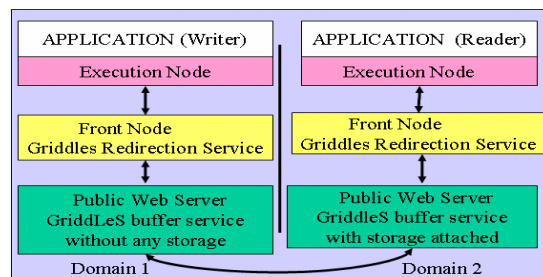


Figure 5. Data redirections between applications.

The redirected data transfer proceeds in two stages: *redirection* and *buffered transfers*. Redirection at the front node passes data from the execution node to the public web server. A buffered transfer from the public web server delivers data to the other domain. In Figure 5, the redirection service on Domain 1 passes data from the writer workflow component to the buffer service, which then performs cross-domain transfers to the other domain. The buffer service on Domain 2 stores the data as it is received from Domain 1. The reader workflow component which is executing on Domain 2 uses the redirection service on the front node to access data from the buffer service. The buffer service on Domain 2 is also responsible for synchronizing between application components. It will also block the reader until data becomes available.

The functionalities of web service client and redirection web services are combined in the implementation of the redirection services. Like a normal web service, it receives a request for redirection and forwards data over to a specific host at a specific port where a buffer service listens for requests on the public web server. The host and port settings may be configured dynamically with a system environment variable. We used the gSOAP [28] toolkit for building this server. The web services and their clients comply with the industry standard WS-Security [29] for secure communications. All server-side components are deployed at web servers under the control of the security administrator and therefore conform to organizational security policies.

4. Experimentation and Results

4.1 The Workflow Experiments

The results presented in this section refer to a real workflow for computing air pollution scenarios from a chain of atmospheric science models. .

In our test case, a global climate model, CCAM (Conformal-Cubic Atmospheric Model) [7], is used to drive the boundaries of a regional climate model, DARLAM (Division of Atmospheric Research Limited Area Model) [7], which in turn feeds data to a

photochemical pollution model, CIT (from the California Institute of Technology) [8]. The scientists explore the effect of different weather patterns on pollution outcomes, and these studies have been applied in atmosphere forecasting and determining air pollution in the atmosphere. Figure 6 illustrates how the three applications, CCAM, DARLAM and CIT, interact in a workflow.

CCAM output data must be fed into DARLAM, but due to mesh refinement, data conversion becomes necessary. The netCDF abstraction layer converts the data according to the individual system which is located under the application layers in both CCAM and DARLAM models. The CCAM and DARLAM run in different mesh resolutions. CC2LAM is a tiny process which sits between CCAM and DARLAM and performs data conversions. It also outputs a much smaller volume of data compared to the input data which is also in netCDF format. Due to network performance, it is convenient to run both CCAM and CC2LAM on the same system as separate processes. GriddLeS is required for synchronization between CCAM and CC2LAM. GriddLeS, as shown via the arrow that runs across the first two organizations, facilitates the redirected data transfer with the second organization, since direct connections are blocked by firewalls. There are also data incompatibilities between DARLAM and CIT which run on separate machines. The DARLAM model produces netCDF format data whereas the CIT model reads and produces standard IEEE format data. A tiny program called LAM2CIT takes the netCDF output of DARLAM and converts it into the IEEE format. Since the CIT model is running on a Linux machine, LAM2CIT needs to run on a Linux machine to produce compatible results. As with CC2LAM, LAM2CIT also produces a much smaller volume of data compared to its input data and is run on a Linux machine at the second organization. GriddLeS also facilitates the redirected data transfer across to the third organization. Despite the firewalls between organizations, GriddLeS successfully provides connectivity on behalf of the applications running on execution nodes. As explained earlier, this is made possible through buffer services on public web servers and lightweight redirection servers at front nodes. Rather than punch holes through firewalls, GriddLeS successfully utilizes ports and protocols that are authorized by existing security policies.

The entire experiment is controlled by a central GNS (Griddles Name Service) [14] web service. GNS makes it possible to identify the nature and location of a file to be accessed. A separate process loads the GNS configuration based on the workflow graph before executing the workflow. The GNS is used for storing file mapping information that describes the

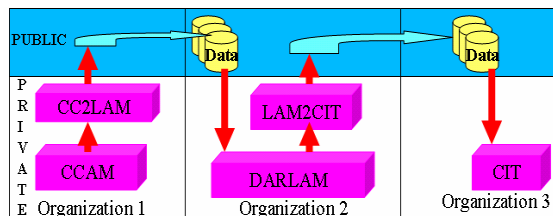


Figure 6. Experimentation data flow diagram

configuration of a particular Grid application. GriddLeS resolves the names and locations of the files when an application tries to open them.

In all the following experiments we explore two scenarios. First, we run all applications sequentially. Second, we start all applications at the same time and run them concurrently. The first scenario is the same as running applications manually and copying files across domains before starting the downstream computation. The second is intuitively better due to the simultaneous execution of certain tasks to save time. In order to select one or the other scenario, we simply place either a PN or SDF director in charge of the workflow, while everything else in the grid remains the same.

4.2 Performance Results

Table 1 presents some performance measurements of the same workflow using different Kepler directors, SDF and PN. With the former, data is written to a local machine and then the entire file is subsequently copied across to a remote machine. In the latter case, all models are started at the same time. Output data is written to a buffer, and is read by the downstream computation from that buffer.

Table 1 Experimentation results

Model Ref.	System Location	Machine Architecture	Accumulated time in seconds	
			SDF	PN
CCAM	CSIRO	NEC SX6	811	832
Cc2lam	CSIRO	NEC SX6	913	832
Darlam	QUT	SGI IRIX	2085	1366
Lam2cit	QUT	Linux	2134	1383
CIT	VPAC	Linux	3696	2124

The pipelined configuration (using the PN director) is faster than the sequential one, as there is an overlap in the execution times between reader and writer applications. The completion time to generate result predictions from the experiments are very similar to previous results [15] where there are no heterogeneity and security restrictions. However, in the present context data flow from one application to another involves a number of redirections in order to resolve restrictions imposed by security policies. In general, the number of redirections will depend on security policies of individual organizations and incompatibilities arising from them. Furthermore, there are data conversions between heterogeneous resources which occur automatically through a netCDF layer. The results indicate that there is no performance impact because of overheads either due to redirections

or due to data conversions. This is because data transfers, subject to GriddLeS intervention through redirection and buffering, are overlapped with execution. Since execution time is more significant than data transfer time, the effects of latency associated with GriddLeS mechanisms are minimized. However the first component execution takes bit longer because of delays in establishing communications channels.

The results illustrate that GriddLeS provides connectivity between applications in a workflow despite firewall restrictions that block direct access. Unfortunately, we cannot compare the performance of the workflow with and without GriddLeS, since connections will not be possible without GriddLeS. However, using the SDF director is the same as running applications manually and copying files across domains before starting the downstream computation. Compared to the SDF director, the PN director minimizes the overhead, since data transfer and execution take place in parallel.

5. Related Work

In the recent past, several workflow demonstrators have been constructed in executing applications across distributed resources [10][11][12][13][28][29][30][31][32][33][34]. None of these involve IO mechanisms as flexible as GriddLeS. Some of them provide simple techniques to deal with a limited class of problems arising from data or security heterogeneity. The Legion file system [35] makes copies of files available before an application runs, as does Nimrod/G [4]. Condor [36], using the Bypass library, allows applications running on remote nodes to access files in their home file system. OGSA-DAI [37] facilitates unified access to data resources such as relational or XML databases and file systems using web services, but the applications require significant modification.

In a similar manner, various platforms such as Globus [38], Condor [36] and SRB [25] cannot handle firewalls in a transparent manner. It is normal for testbeds using these systems to require a range of ports to be kept open. Various general-purpose “firewall evasion” solutions do exist, primarily for typical Internet utilization, e.g., web surfing, email or instant messengers. SOCKS proxies [39] allow web browsers or file transfer protocol (FTP) clients to access data through third-party proxies. Port forwarding via SSH will tunnel connections through from a number of local ports to associated destinations to which the local ports are mapped. Our group has also developed a rerouting and multiplexing system, REMUS [40] that reroutes connections through tunnels and proxies. There are a number of other projects that similarly use tunnels and

proxies [41], but those methods are not as holistic and transparent as GriddLeS is in providing a general-purpose flexible IO mechanism.

6. Conclusions

In this paper, we identified the adverse impact of heterogeneity that arises from architecture differences and inconsistent security policies between resources. Workflow components are unable to communicate when connections are blocked or when incompatible data formats are used. We then presented GriddLeS, which resolves such problems through a flexible IO infrastructure that can use several mechanisms to enable communication between workflow application components. We then demonstrated GriddLeS using experiments with a real workflow in an atmospheric science discipline. Our experiments demonstrate the ease in which we can combine secure heterogeneous resources through existing abstractions without compromising individual security policies. The results also show that, for a special class of problems where execution time is more significant when compared to data transfer time; latency from various sources can be completely hidden. This was despite a number of data redirections and conversions between source and destination machines. As a whole, the GriddLeS runtime environment hides network-related complications and allows application scientists to run applications seamlessly over a number of secure heterogeneous resources. In previous papers, we had proven the success of GriddLeS in homogeneous environments, where the firewall restrictions are very minimal [13][14][15]. The results presented in this paper establish the usefulness of GriddLeS in secure heterogeneous environments.

Acknowledgements

The GriddLeS project is supported by the Australian Research Council and Hewlett Packard under an ARC Linkage grant, and the Australian Government Department of Communications, Information Technology and the Arts under a GranetNet grant.

We would like to thank our colleagues, John McGregor, Jack Katzfey and Martin Dix from the CSIRO Division of Atmospheric Sciences, who provided the models used in this experiment. Our special thanks to Tom Peachey for proof reading this manuscript. We are grateful to Kepler and Ptolemy development teams for their support.

References

- [1] Foster, I., Kesselman, C., (eds.): The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, USA (1999).
- [2] Foster, I., Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, Int'l J. of Supercomputer Applications, vol. 11, no. 2 (1997) pp. 115-128.
- [3] The Globus Toolkit: <http://www-unix.globus.org/toolkit/>
- [4] Abramson, D., Giddy, J., Kotler, L.: High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid, In Int'l. Parallel and Distributed Processing Symposium (IPDPS), Cancun, Mexico, May 2000. <http://www.csse.monash.edu.au/~david/nimrod/>.
- [5] e-Science Grid Environments Workshop, e-Science Institute, Edinburgh, Scotland, May 2004, <http://www.nesc.ac.uk/esi/events/>.
- [6] Scientific Data Management Framework Workshop, Argonne National Labs, August 2003. <http://sdm.lbl.gov/~arie/sdm/SDM.Framework.wshp.htm>.
- [7] McGregor, J. L., Nguyen, K. C., Katzfey, J. J.: Regional climate simulations using a stretched-grid global model. In: Research activities in atmospheric and oceanic modeling, H. Ritchie (ed.). (Report; 32; WMO/TD - no. 1105) [Geneva]: WMO. (2002) 3.15-3.16
- [8] Tory, K.J., Cope, M.E., Hess, G.D., Lee, S.H., Wong, N.: The use of long-range transport simulations to verify the Australian Air Quality Forecasting System. 14th annual BMRC Modelling Workshop, A. J. Hollis, and P. J. Meighen (editors) (Report, 90) Bureau of Meteorology Research Centre. (2002) pp. 19-23.
- [9] Kepler Project <http://kepler-project.org>
- [10] Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludäscher, B., Mock, S.: Kepler: Towards a Grid-Enabled System for Scientific Workflows, in the Workflow in Grid Systems Workshop in GGF10 - The 10th Global Grid Forum, Berlin, March 2004.
- [11] Taverna Project: <http://taverna.sourceforge.net>
- [12] Yu, J., Buyya, R., A Taxonomy of Workflow Management Systems for Grid Computing, Technical Report GRIDS-TR-2005-1, Univ of Melbourne (2005). <http://www.gridbus.org/reports/GridWorkflowTaxonomy.pdf>.
- [13] Abramson, D., Kommineni, J., McGregor, J. L., Katzfey, J.: An Atmospheric Sciences Workflow and its Implementation with Web Services, Future Generation Computer Systems, The Int'l J. of Grid Computing: Theory, Methods and Applications, vol. 21 (2005) 69-78.
- [14] Kommineni, J., Abramson, D.: Building Virtual Applications for the GRID with Legacy Components, European Grid Conference, held at Science Park Amsterdam, The Netherlands, February 14-6 (2005).

- [15] Abramson, D., Kommineni, J., Altintas I.: Flexible IO services in the Kepler Grid Workflow system, Int. Conference on eScience and Grid Technology, Dec. 5 – 8 (2005) Melbourne.
- [16] Ho, T. and Abramson, D. “The GriddLeS Data Replication Service”, IEEE Conference on e-Science and Grid Computing, Melbourne, Dec 2005.
- [17] Unidata <http://www.unidata.ucar.edu/software/netcdf/>
- [18] XML in 10 points <http://www.w3.org/XML/1999/XML-in-10-points.html>
- [19] Nec SX6 Multi-node http://www.hpccc.gov.au/facilities/SX-6_MultiNode.pdf
- [20] Sirius (SGI Origin 3000) <http://www.its.qut.edu.au/hpc/hardware/sirius.jsp>
- [21] Brecca Linux Cluster http://www.vpac.org/content/systems_and_support/facility/linux_cluster.php
- [22] Ptolemy II, <http://ptolemy.eecs.berkeley.edu/ptolemyII/>
- [23] Ylonen, T. and Lonvick, C.: SSH Protocol Architecture. Internet Draft, Network Working Group, The Internet Society, December 2004.
- [24] Allcock, W., Bresnahan, J., Kettimuthu, R., Link, M. ., Dumitrescu, C., Raicu, I., Foster, I.: The Globus Striped GridFTP Framework and Server, Proceedings of Super Computing 2005 (SC05), November (2005).
- [25] SRB <http://www.sdsc.edu/srb/index.php/FAQ>
- [26] Design, Performance and Scalability of a Replica Location Service http://www.globus.org/toolkit/presentations/GlobusWorld_2005_Session_6c.pdf
- [27] Gfarm file system <http://datafarm.apgrid.org/>
- [28] van Engelen, R. The gSOAP toolkit 2.0., Technical Report, <http://www.cs.fsu.edu/~engelen/soap.html>.
- [29] OASIS Web Services Security http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- [30] “Web Services Made Easier: <http://java.sun.com/webservices/jwsdp/index.jsp>
- [31] Apache Axis Soap toolkit – Web Services, <http://ws.apache.org/axis/>
- [32] http://www.cogkit.org/release/4_1_3/
- [33] I. Taylor, M. Shields, and I. Wang. Resource Management of Triana P2P Services. Grid Resource Management, Kluwer, Netherlands, June 2003.
- [34] S. McGough et al. Workflow Enactment in ICENI. In UK e-Science All Hands Meeting, Nottingham, UK, IOP Publishing Ltd, Bristol, UK, Sep. 2004; pp. 894-900.
- [35] White, B., Grimshaw, A., Nguyen-Tuong, A.: Grid-Based File Access: The Legion I/O Model. Ninth IEEE Int. Symposium on High Performance Distributed Computing, Pittsburgh, Pennsylvania, Aug 1-4, 2000.
- [36] Condor Firewall Requirements, <http://www.escience.cam.ac.uk/projects/camgrid/firewall.html>, 2005
- [37] K. Karasavvas, M. Antonioletti, M.P. Atkinson, N.P. Chue Hong, T. Sugden, A.C. Hume, M. Jackson, A. Krause, C. Palansuriya. Introduction to OGSA-DAI Services. Lecture Notes in Computer Science, Volume 3458, Pages 1-12, May 2005.
- [38] Von Welch, Globus Toolkit Firewall Requirements, <http://www.globus.org/toolkit/security/firewalls/Globus%20Firewall%20Requirements-7.pdf>, NCSA/U. of Illinois, 2005.
- [39] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and Jones, L.: SOCKS protocol v5. RFC 1928, 1996.
- [40] J. Tan, D. Abramson, and C. Enticott. Bridging organizational network boundaries on the grid. In Proc. of the 6th IEEE/ACM International Workshop on Grid Computing (Grid 2005), Seattle, Nov. 13–14 2005.
- [41] S. Graupner and C. Reimann. Globus grid and firewalls: issues and solutions in a utility data center environment. Technical report HPL-2002-278, Hewlett-Packard Company, October 2, 2002.