UK e-Science Technical Report Series ISSN 1751-5971

Towards tractable toolkits for the Grid: a plea for lightweight, usable middleware.

Jonathan Chin <jonathan.chin@ucl.ac.uk> Peter V. Coveney <P.V.Coveney@ucl.ac.uk>

Centre for Computational Science
Department of Chemistry
University College London
Christopher Ingold Laboratories
20 Gordon Street
London WC1H 0AJ
United Kingdom
http://www.chem.ucl.ac.uk/ccs/

Abstract

In the light of our ongoing attempts[1] to solve real scientific problems using heterogeneous high-performance computing equipment linked over a Grid, we discuss several significant barriers to widespread acceptance of Grid technology through the end-user community of application scientists, and propose some solutions.

Executive Summary

We have attempted to make use of Grid technologies for serious scientific work, as part of the EPSRC-funded RealityGrid[2] e-Science pilot project. We have encountered serious middleware-related problems which are hindering scientific progress with the Grid:

- The existing toolkits have an excessively heavy set of software and administrative requirements, even for relatively simple demands from applications.
- Existing toolkits are painful and difficult to install and maintain, due to excessive reliance on custom-patched libraries, poor package management, and a severe lack of documentation for end-users.
- Existing standards bodies and the task forces within the UK e-Science programme are not engaging sufficiently with the applications community, and run a substantial risk of producing and implementing Grid architectures which are irrelevant to the requirements of application scientists.

We argue that it is important to develop a simple, lightweight Grid middleware which is "good enough" for rapid adoption, rather than taking longer to develop a solution which will, supposedly, suit all needs. Such a toolkit must be:

- substantially more portable, lightweight, and modular in design
- produced in very close collaboration with application scientists
- sufficiently well-documented that end-users will be able to port existing codes to use Grid techniques with the minimum of hassle.

Introduction

In recent years, significant effort has been invested in Grid technologies[3], often touted as the future of scientific computing[4], from the low-end perspective of harvesting cycles from idle workstations to the high-end perspective of allowing high-performance computing resources to be shared effectively across geographically-distributed Virtual Organizations[3].

Grid development in the UK has been primarily driven by the UK e-Science initiative. This consists of, inter alia, many e-Science pilot projects, developing applications to use Grid technologies for advancement of scientific research, and a Core Programme[5], tasked with supporting the pilot projects, and advancing the development of Grid middleware in collaboration with industry. It is important to recognize that the Core Programme is a subset of the eScience programme as a whole.

This document highlights some of the experiences we have been through over the past two years as part of RealityGrid[2], an EPSRC pilot project and part of the UK e-Science programme, with a remit to deploy, develop, and extend Grid technologies for modelling of condensed matter and materials, and most recently, as part of the award-winning[6]TeraGyroid [1] project, where unprecedented levels of supercomputing power were harnessed together across two continents to perform what are thought to be the largest lattice-Boltzmann fluid-dynamical simulations ever performed. These projects have unearthed several strong (and, we feel, not uncommon) criticisms of the available Grid toolkits, which are discussed below. By far the most common toolkit employed for scientific work is Globus Toolkit 2[7], although version 3[8] is now coming into use. Our criticisms will be primarily levelled at the Globus toolkit (both versions 2 and 3 unless otherwise specified), although they are directly relevant to other middleware systems as well.

It is important to bear in mind that a Grid is only as useful as the resources it connects together. To be of scientific use, therefore, it is critical that as many diverse groups of scientists as possible begin to use the Grid; we feel that the issues we discuss in this paper are preventing uptake and deployment of Grid technologies outside of specialized, Grid-centric projects: such barriers are harmful to the progress of the Grid as a whole.

Our article differs from most of the others so far written on Grid middleware in that the authors are "applications" scientists who are expected to make use of such software, as opposed to computer scientists or software engineers with a vested interest in its development and widespread deployment. We have written this paper in the hope that it will encourage much closer interactions between Grid software engineers and the user community in the future.

User-unfriendly toolkits

Bloat

In order to develop codes and methodologies for working with the Grid, end-users must be able to rapidly build prototype applications, and the speed and ease with which they can do so varies inversely with the complexity of the toolkit available.

Our principal criticism is that existing toolkits are far too large and cumbersome: many are monolithic or effectively monolithic pieces of software, which take a substantial effort to install, even for client-side code; worse, this effort must often be on the part of system administrators, rather than ordinary users. There are many contributing factors to this.

The term "Grid middleware" is often used to describe a whole suite of software for authentication, encryption, resource discovery and metadata handling, job and queue management, communication via remote procedure calls (RPC) and file transfer, and many related tasks. However, a monolithic system attempting to provide all of these facilities in one unit will be cumbersome to deal with for the user who only wishes to deploy a small part of that functionality. This will often be the case for a first-time user, who wishes to try the system out on a small scale.

For certain features, Globus Toolkit version 3 requires or strongly recommends the installation of additional heavyweight packages such as a relational database, or a servlet container package, contributing further to the administrative burden.

Existing software is not re-used properly

Dependencies are a significant problem, particularly with the Globus toolkit. Initially, it would seem laudable that the toolkit builds on existing open-source libraries such as OpenSSL[9], but it does so by requiring modified, non-mainstream versions of these libraries. It is not possible to install Globus on top of the standard, vendor-supplied copy of OpenSSL; instead, one must build Globus' own version, so that the open-source library

is duplicated, rather than re-used. The result is that systems administrators must maintain multiple different installations of certain libraries on any given machine. Moreover, if any changes are made to the mainstream version of a library, they must be "backported" to the toolkit-modified version. If a security problem is found in the mainstream version, end-users of the Grid toolkit are left running insecure systems until the security fix has been backported.

Globus now bundles the required modified libraries along with the rest of the toolkit, but this simply increases the effective size of the toolkit, without solving any of the other problems: a binary install of Globus Toolkit 3 on an Intel-based workstation takes upwards of 100 megabytes.

Reliance on modified third-party software also presents significant obstacles to deployment of Grid middleware on less common platforms and architectures; this makes it much harder to make specialized computing facilities available as Grid services.

The world does not need another package manager

There have been attempts to solve the problems of software dependencies and multiplatform deployment by introducing a separate packaging system (Globus Packaging Tool, or GPT) for the middleware toolkit -- leaving aside the fact that it's yet further bloat, it can make things even worse! Any given UNIX workstation will probably have a large amount of software installed using the OS vendor's packaging system, along with custom, site-specific software, possibly organized using a local-package system such as GNU stow[10]. In addition, many languages such as Perl or Python have their own system for installing language-specific libraries. Adding yet another packaging system (for the sole purpose of installing a middleware library) will only complicate life further and render it even more difficult to make the middleware cooperate with existing software. For example, our experience has been that the GPT will attempt to install out-of-date versions of Perl modules without even checking if any version at all has already been installed, let alone a more recent version. Maintaining multiple package installations can be difficult enough, let alone doing so with multiple package managers. Consistency with existing package management systems would require either that the middleware installation system can be configured to use vendor packaging systems (in the same way that, for example, Perl and Python modules can be installed as Red Hat packages for some linux distributions), or that a substantial amount of effort be invested to make the package management aware of other systems in order to avoid conflicts.

OGSI wrappers make life harder

The Globus Toolkit version 3 provides much of the GT2 functionality through an OGSI-compliant interface. Currently, it does not do so natively, and the current stopgap solution is to encapsulate the old functionality through the use of Java-based wrappers. While this may work towards a very welcome level of interoperability with other middleware systems, it does nothing about the problems enumerated above; it merely adds to the large bolus of software to be installed.

Administrative overhead

Once the middleware is installed, it must be configured. Besides the many configuration files to be edited, there is a substantial amount of bureaucratic overhead required to set up user certificates, and the appropriate mappings on each machine involved. On the rapid timescales required for the TeraGyroid[1] project, these inevitable issues were dealt with by keeping lists of users on a private editable webpage, or wiki[11]; a more straightforward tool for such virtual organization management might have simplified matters further. A substantial problem is that authentication in Globus Toolkit versions 2 and 3 relies on static grid map files, which must be updated on all machines used in a project, often manually, due to lack of deployment of any more useful tool to do so.

It appears to be taken for granted by the authors of the existing middleware that many different remote users will want to perform tasks on a given machine; the gatekeeper program will run as a super-user, in order to have access to other user accounts and privileges. This is a reasonable assumption to make for large machines; it is not so reasonable for a workstation or small machine offering a single service or testbed. The ability to run a small installation of the middleware, running as a single (but authenticated) user and offering a single service, not necessarily relying on the full might of the middleware, would be of significant benefit to those who only wish to try out Grid techniques, or to deploy them on low-end systems.

Poor documentation

A default installation of the Globus toolkit comes with exceedingly little documentation; the packaging tool comes with manual pages, but there is no corresponding documentation for the large suite of tools which it packages; a directory to contain documentation automatically generated from the source code is left empty. Installation documentation is sparse, and it's not uncommon for administrators to have to resort to searching the web for installation hints.

API documentation now exists on the web, although it often assumes that the end-user is familiar with Grid methodology. This is fine for the specialized, Grid-centric community, but if the Grid is to take off as a common mode of operation amongst computational scientists, then there is an urgent need for documentation and tutorials targeted at an application scientist who is new to the field; the lack of such documentation will, at best, slow uptake, and at worst, scare off potential users.

Issues with OGSI and OGSA

The criticisms we have made so far are principally levelled at middleware implementations, but we have some additional concerns about the OGSA and OGSI standards, to which many implementations are now aiming to conform.

It is of concern to us that OGSI modifies, rather than re-uses, existing Web Services frameworks such as WSDL. WSDL is an accepted and widely deployed industry standard; adding more parts to its specification means that OGSI implementations cannot simply lift and re-use existing code, but must either modify what already exists, or start from scratch.

The difficulties enumerated above are specific instances of the more general problem that while systems we've tried permit the solution of large, complex problems (through correspondingly complex designs), these may not scale down to provide simple solutions to simpler problems.

If the complexity of a Grid lies at the middleware level, then it can be frustratingly difficult for end-users to start off by building small systems. However, if it were made as easy as possible to create Grid services, or to expose existing components (such as computational codes, visualization engines, databases, or instruments) as Grid services, then the complexity can then lie at the level of creating a system from a composition of these basic services: simple problems can have simple solutions created by joining together a small number of components, and complex problems can be solved through correspondingly larger numbers of components. Moreover, this approach would then lower the entry barrier for Grid deployment, allowing quicker uptake of the technology.

Targeting the wrong problem field?

It has already been noted by other groups[12] that much more documentation is required for existing middleware toolkits. It is of note that much of the existing demonstration code is written in Java, as are many of the examples available to first-time users. Despite its many qualities, Java is not the language of choice for the scientific computing community, which still relies on a very substantial codebase written in C, C++, and Fortran. Illustrative examples in Java may be of substantial pedagogical value, but they are of little use to someone who has a Fortran code they wish to deploy (in a steerable fashion) across the Grid, and who may have little experience of the (often subtle and complicated) process of binding a library to a different language. We maintain that a successful middleware must have bindings to multiple languages.

Many scientific and industrial applications will require more than just job submission and remote procedure call functionality to work effectively across a Grid; substantial amounts of binary data must also be transferred. It is of considerable concern that this issue may not be fully addressed by an underlying Web Services framework, which is more oriented to the demands of the business community.

Poor engagement with user community

The UK e-Science Core Programme is, at present, mostly composed of software engineers, computer scientists, and distinguished individuals from the applications community, of whom very few, if any, are likely to have to interact with Grid middleware on a day-to-day basis. This being the case, it is perhaps not too surprising that the

Grid infrastructure is evolving in a way unsuited to the demands of the end users: ordinary scientists who wish to deploy their software across the Grid. In fact, this shortcoming was pointed out two years ago by Allan *et al*[13]. We feel that it is of significant concern not only that the Grid is, in its present state, almost unusable for the serious work of application scientists, but that it is evolving in a way which seems likely to encourage patterns of operation which are inappropriate for their needs.

In addition, there is a similar domination of computer scientists and lack of end-users or real engagement with the applications community in the Global Grid Forum: this can only be harmful in a standards body, if it is ever to produce standards of any utility. We note in passing that the Message Passing Interface (MPI) standard was developed over a two-year intensive process, followed by the rapid and widespread emergence of usable implementations; Grid standards have now been under discussion for more than three years, with little to show to the end-users.

Allan *et al* pointed out that "Take-up by application users is however still slow as it superficially offers little that cannot be done using SSH or another mechanism" -- from the point of view of application scientists, this is still true, and substantial effort needs to be made for the two communities to re-engage.

Experiences with certain existing toolkits

Our experiences with the Grid have primarily been focussed around two pre-existing applications for simulations of complex fluids using lattice-Boltzmann methods in two and three dimensions. The applications, LB2D and LB3D, are both portable scientific codes which, it appeared, would benefit highly not only from the ability to be deployed transparently across a Grid, but also from the more long-term benefits which Grid middleware ought to facilitate, such as computational steering and distributed visualization and data management. LB3D is an extremely scalable code, placed in the highest performance class on the UK's HPCx[14] machine, and has been deployed on a wide range of supercomputing architectures. LB2D is a versatile workstation-class code which can also perform task-farmed calculations across a similarly wide range of machines. Some of the advantages we hoped to be able to obtain from Grid-enablement of the codes included transparent remote execution and steering of codes, and the ability to easily create calculation systems by connecting several Grid Services together (for example, connecting a lattice Boltzmann component to another which performs visualization). We emphasise that we are looking for more than just compute-centric remote job submission to HPC resources: data-centric steering and visualization, and community-centric collaborative systems are also important to us.

LB3D was Grid-enabled using the RealityGrid[2] steering library, which in turn uses Globus for resource discovery and communication. Custom client code was written to permit remote job submission and computational steering of the code[15].

Despite vast amounts of effort within the RealityGrid project being applied to Grid-enablement of the code within the RealityGrid project, after two years of work, the state of the Grid is still such that there are no significant benefits offered to application scientists, who currently consider it easier (and more reliable) to submit and control tasks through more traditional means such as SSH. In order to deploy the code across the Grid, it was necessary to write separate scripts for each machine on which it would be run -- a lengthy and laborious process, and, sometimes much more complicated than manual installation of the code through traditional means. This tedious requirement has been noted by others, such as Butchart *et al*[16].

Even simple utilities such as the `globus-url-copy` file transfer program are spurned, since firewall rules at different sites often complicate its invocation, and because of user interface issues such as its insistence on the use of the `file://` URL scheme for local files, and inability to copy multiple files in a single invocation.

As an experiment in the use of alternative middleware designs, LB2D was interfaced to the `OGSI::Lite`[17] toolkit, written by Mark McKeown at the University of Manchester. `OGSI::Lite` is a container to allow simple creation of Grid services, written using the Perl language. In order to be made available as a Grid service, a software component needs only to be placed in a specific location, and inherit from a supplied base class. It turned out to be possible to build a simple remote job submission and steering system using LB2D and `OGSI::Lite` with a few evenings' work, by writing a small wrapper module which provided an interface between the Grid service container and the actual LB2D code; details are available on the web[18].

We attribute this simplicity of operation to several key aspects of `OGSI::Lite`. Firstly, it solves one problem, and solves it well: the problem of exposing programs as a Grid service. Security, authentication, RPC, and many

other issues are handled by third-party software. Secondly, dependencies on third-party software are straightforward: it will work with vendor-shipped versions of the OpenSSL library to provide transport-layer security; it re-uses unmodified third-party modules to handle XML parsing and a large number of other tasks. Thirdly, because of the straightforward dependencies, it will interoperate with existing package management systems without any abnormal administrative overhead. Fourthly, the manner in which it builds on existing implemented standards (such as SOAP for remote messaging) permitted us to launch and steer simulations using only an unmodified version of the Mozilla[19] web browser which supported SOAP messaging -- specialized client software was not required! The ability to perform such a task, not originally envisaged by the designers, is a strong indication of well-designed software.

There is currently a project called DL_MESO[20] taking place within the CCP5 consortium at the Daresbury Laboratory to construct a general-purpose mesoscale modelling package, very similar in certain aspects to the LB2D and LB3D codes mentioned above, and an ideal opportunity to disseminate Grid techniques. However, despite this work taking place at one of the UK's main e-Science development centres, and despite encouragement on our part, little interest has been shown towards designing the code with the Grid in mind. This is not at all surprising: it is hard to convince anyone that investing already scarce development resources in Grid-enablement will pay off, given the current state of the middleware.

Towards tractable toolkits

While, as application scientists, we have encountered significant obstacles to the uptake of Grid technology, the situation is not impossible; however, we strongly believe that a substantially different approach to middleware is required.

Tolerance of heterogeneity

To create a Grid out of heterogeneous systems, one must be able to **easily** deploy the appropriate middleware on these systems; a build system so complicated that most users install from binary packages leaves little hope for those who wish to compile from source on new platforms. This can be a particular problem for high-end or cutting-edge machines of the sort typical in scientific research, which often have highly-specialized configurations.

Interoperability also means decreasing the number of dependencies and custom-patched libraries required, permitting re-use of existing libraries. We emphasize that adding wrapper layers is not a solution: while it can facilitate interoperation with other OGSI-compliant systems, it still increases footprint and dependency tree size, making deployment across heterogeneous systems more difficult.

Simple grid containers are needed for end users

A particular success of the `OGSI::Lite` toolkit is the ease with which new grid containers may be constructed for existing application codes. A straightforward, well-documented method for applications scientists to expose their existing codes as Grid services would, we feel, provide huge benefit to the Grid community as a whole, from the resulting increase in uptake. In particular, this will mean development of hosting environments for codes written in C, C++, or Fortran, as opposed to the current focus on Java.

Orthogonalization and factorization

Perhaps the easiest way to reduce the minimum footprint of a middleware system is to separate out its components in such a way that unused facilities do not have to be installed, and an application scientist wishing to implement a Grid service need only deal with those pieces of the middleware which are relevant to the service, without worrying about breaking compatibility with the rest of the Grid. Delegation of functionality to off-the-shelf, unpatched versions of Open Source libraries would also vastly increase ease of installation and use.

Much of the proposed Open Grid Services Infrastructure extends Web Services frameworks to provide statefulness, factories, and other useful facilities. While the layered structure of OGSA and OGSI could facilitate factorization, we strongly suggest that altering existing frameworks (by, for example, altering rather than

re-using the already widely-adopted WSDL standard) would have quite the opposite effect. There have now been proposals[21] for implementing such facilities on top of existing Web Services, with the explicit aims of simplicity and minimalism, re-using existing tools and practices wherever possible.

Better communication with end-users

We believe that many of the problems listed above result from lack of communication between the infrastructure-development community and the end-user community. Not only should middleware development be more focussed on lightweight, portable systems, but it should be developed in much closer collaboration with application scientists, possibly to the extent of middleware developers being "embedded" with research groups (or *vice versa*) during the design and construction of such software. To see this lead to successful results or products would perhaps take a full-time software engineer a year, during which the lightweight-style approach to middleware is implemented in a more comprehensive and robust manner. We suggest that, given the success of the `OGSI::Lite` approach, it should be pursued further.

Such a change of direction must be pursued with real urgency; otherwise there is a risk that user uptake will remain low and begin to wane. One route among several for doing this might be via the now established Open Middleware Infrastructure Institute (OMII), part of the UK eScience Core Programme.

Lightweight OGSI implementations: a way forward?

While we do not believe that simply wrapping existing toolkits in an OGSI-compliant layer is a reasonable solution, success may lie in the direction of OGSI implementations. It is not immediately clear that some of the techniques encouraged by the OGSI structure map well onto scientific problems, but this is the sort of issue that can only be solved through closer collaboration and dialogue with application scientists.

Existing implementations of OGSI are still somewhat incomplete, and there is still substantial ongoing discussion concerning interpretation of various parts of the OGSI specification. However, it may be possible to implement parts of the OGSI spec using existing, unmodified Web Services tools. The Open Grid Services Architecture is a layered design -- we urge that implementors attempt to separate implementations of each layer as much as possible, possibly through construction of "profiles" or presentation layers which provide only the minimum required functionality.

Conclusions

While trying to perform serious scientific tasks, both supercomputer- and workstation-class, using Grid technologies, we encountered very significant barriers imposed by the currently available middleware. We attribute this to the size, monolithic nature, and poor code re-use of the systems we used. Important progress was made using a lightweight implementation of part of the OGSI specification, and we believe that this points the way for further development, emphasising the value of leveraging existing infrastructure such as Web Services.

The case has been made before (most notably by Richard Gabriel in his essay " Worse is Better[22]") that in large software projects, it is more important to build a straightforward, "good enough" solution which will gain rapid uptake from the user community, than to develop a supposedly perfect all-purpose system in isolation. Ease of installation and facility with which legacy codes can be ported to the Grid environment are critical to user uptake: a "Resilient, Reliable, Robust" Grid is of little help to anyone unless it is actually usable.

Acknowledgements

We are grateful to Mark McKeown of Manchester University, Matt Harvey of UCL, Malcolm Atkinson and Dave Berry of the National e-Science Centre, and Tony Hey, Director of the UK e-Science Core Programme, for useful and enlightening discussions and criticism.

Appendix: A response to WSRF

Since this paper was originally written, the Web Services Resource Framework(WSRF)[23] was announced. WSRF can be regarded as a "straightforward refactoring of the concepts and interfaces developed in the OGSI V1.0 specification in a manner that exploits recent developments in Web services architecture"[24]. We feel that the WSRF specification represents a significant step forward for the possibility of useful middleware, and addresses many of the points raised above.

Specifically, the monolithic OGSI specification has now been subdivided into many smaller parts, which do not necessarily have to be implemented together. Partly as a result of this, the specification is much less prescriptive about modes of operation: whereas the OGSI specification strongly encouraged the use of the Factory pattern for creating Grid services, many ways are possible under WSRF. Moreover, should any single component of the framework be found to be ill-suited to a particular application, it can be replaced without affecting other components. Another very significant difference is that it builds on top of existing specifications such as WSDL, rather than requiring modifications: this should allow use of unmodified, off-the-shelf tooling, and significantly improve compatibility problems.

Of course, a useful specification is only the first step towards building middleware which can be deployed fruitfully in scientific applications; many of the criticisms made above were more related to implementation details, and these must still be borne in mind. The approach of wrapping up an existing middleware toolkit in a WSRF-compliant layer will still have all the old problems with installation, administration, and documentation: it will simply be able to interoperate with other toolkits through WSRF. We feel that now is the ideal time for development of lightweight implementations of WSRF, preferably in close communication with the end-user applications community, and that this represents the optimal path towards realizing a Grid of widespread deployment and utility.

References

[1] The TeraGyroid Project , http://www.realitygrid.org/TeraGyroid.html
[2] The RealityGrid project , http://www.realitygrid.org/
[3] The Anatomy of the Grid , Ian Foster , Carl Kesselman , Steven Tuecke , (2001) http://www.globus.org/research/papers/anatomy.pdf
[4] The Grid : Blueprint for a New Computing Infrastructure , Ian Foster , Carl Kesselman , (1998) Morgan Kaufmann http://www.amazon.com/exec/obidos/ASIN/1558604758/o/qid%3D958665349/sr%3D2-1/102-0153426-8841738
[5] The UK e-Science Core Programme and the Grid , Tony Hey , Anne E. Trefethen , Future Generation Computing Systems **18** p 1017(2002) http://www.rcuk.ac.uk/escience/documents/CoreProgGrid.pdf
[6] TeraGyroid wins HPC Challenge , The RealityGrid project , (2003) http://www.realitygrid.org/news.html
[7] Globus Toolkit 2.4 , The Globus Alliance , (2003) http://www.globus.org/gt2.4/
[8] Globus Toolkit 3.0.2 , The Globus Alliance , (2003) http://www-unix.globus.org/toolkit/download.html
[9] The OpenSSL Project , http://www.openssl.org/
[10] GNU Stow , http://www.gnu.org/software/stow/
[11] WikiWikiWeb , Portland Pattern Repository , http://c2.com/cgi/wiki
[12] UK Experience with OGSA: report of the workshop held at CCLRC , Dave Berry et al , (2003) http://www.nesc.ac.uk/technical_papers/UKeS-2003-02.pdf
[13] Evaluation of Globus and Associated Grid Middle-ware , R. J. Allan , D. R. S. Boyd , T. Folkes , C. Greenough , D. Hanlon , R. P. Middleton , R. A. Sansum , (2001) CLRC e-Science Centre http://www.hpcvc.lboro.ac.uk/eval_clrc.pdf
[14] HPCx , The HPCx Consortium , http://www.hpcx.ac.uk/
[15] Steering in computational science: mesoscale modelling and simulation , J. Chin , J. Harting , S. Jha , P. V. Coveney , A. R. Porter , S. M. Pickles , Contemporary Physics **44** p 417-434(2003)
[16] OGSA First Impressions: A Case Study using the Open Grid Service Architecture. , B. Butchart , C. Chapman , W. Emmerich , In S. Cox (ed): Proceedings of the UK E-Science All Hands Meeting, Nottingham p 810-816(2003) http://www.cs.ucl.ac.uk/staff/w.emmerich/publications/AllHands03/ogsa.pdf
[17] OGSI::Lite - an OGSI implementation in Perl , Mark McKeown , (2003) http://www.sve.man.ac.uk/Research/AtoZ/ILCT
[18] Adventures with LB2D and OGSI::Lite , Jonathan Chin , (2003) http://the.earth.li/~jon/work/ogsi/writeup/
[19] The Mozilla Project , http://www.mozilla.org/

[20] DL_MESO: A mesoscopic simulation package , CCLRC Daresbury Laboratory , (2003) http://www.ccp5.ac.uk/dlmeso/DL_MESO.html
[21] A Grid Application Framework based on Web Services Specifications and Practices , Savas Parastatidis , Jim Webber , Paul Watson , Thomas Rischbeck , (2003) North East Regional e-Science Centre http://www.neresc.ac.uk/ws-gaf/
[22] The Rise of "Worse is Better" , Richard P. Gabriel , http://www.jwz.org/doc/worse-is-better.html
[23] Web Services Notification and Web Services Resource Framework , IBM developerWorks , http://www-106.ibm.com/developerworks/webservices/library/ws-resource/
[24] The WS-Resource Framework , The Globus Alliance , IBM , HP , http://www.globus.org/wsrf/

$Id: lgpaper.xml,v 1.20 2004/02/04 19:21:18 jon Exp $