

Applications Development for the Computational Grid

David Abramson

Faculty of Information Technology
Monash University,
Clayton, Vic, Australia, 3800
davida@csse.monash.edu.au
<http://www.csse.monash.edu.au/~davida>

Abstract. The Computational Grid has promised a great deal in support of innovative applications, particularly in science and engineering. However, developing applications for this highly distributed, and often faulty, infrastructure can be demanding. Often it can take as long to set up a computational experiment as it does to execute it. Clearly we need to be more efficient if the Grid is to deliver useful results to applications scientists and engineers. In this paper I will present a raft of upper middleware services and tools aimed at solving the software engineering challenges in building real applications.

1 Introduction

e-Science, enabled by the emerging Grid computing paradigm [28], tightly couples scientists, their instruments (e.g., telescopes, synchrotrons, and networks of sensors), massive data storage devices and powerful computational devices. This new discipline allows scientists to interact efficiently and effectively with each other, their instruments and their data, even across geographic separations, thereby ameliorating the tyranny of distance that often hinders research. Data can be captured, shared, interpreted and manipulated more efficiently and more reliably and on a far greater scale than previously possible. Data can be presented for interpretation in new ways using scientific visualization techniques and advanced data mining algorithms. These new technologies enable new insights to be derived and exploited. The data may also drive simulation models that support prediction and “what-if” analyses. The models and their results may be archived for later use and analysis, and shared securely and reliably with scientific collaborators across the globe. The resulting network of people and devices is empowered to interact more productively and to undertake experiments and analyses that are otherwise impossible.

In spite of tremendous advances in middleware and internet software standards, creating Grid applications that harness geographically disparate resources is still difficult and error-prone. Programmers are presented with a range of middleware services, a raft of legacy software tools that do not address the distributed nature of the Grid, and many other incompatible development tools that often deal with only part of the Grid programming problem. So, a scientist might start with an idea for an innovative experiment but quickly become distracted by technical details that have little

to do with the task at hand. Moreover, the highly distributed, heterogeneous and unreliable nature of the Grid makes software development extremely difficult. If we are to capitalize on the enormous potential offered by Grid computing, we must find more efficient and effective ways of developing Grid based applications.

2. Software engineering for the Grid

A critical ingredient for success in e-Science is appropriate Grid-enabled software which, to date, has lagged behind the high-performance computers, data servers, instruments and networking infrastructure.

All software follows a lifecycle, from development through execution, and back again, (Figure 1). Grid software is no exception, although there are sufficient differences in the details of the various phases in the lifecycle to make traditional tools and techniques inappropriate. For example, traditional software development tools rarely support the creation of virtual applications in which the components are distributed across multiple machines. In the Grid, these types of virtual applications are the norm. Likewise, traditional methods of debugging software do not scale to the size and heterogeneity of infrastructure found the Grid. Here, we identify four distinct phases of importance, development, deployment, testing and debugging and execution.

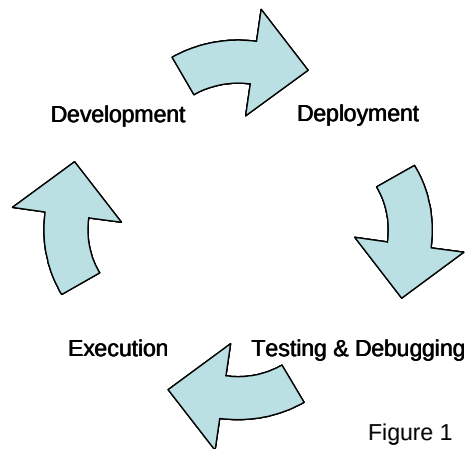


Figure 1

2.1 Development

Initially, software is developed using the most appropriate tools and programming languages for the task at hand. The process involves the specification, coding and compilation of the software. In the Grid, there is a very strong focus on building “virtual applications”, or workflows, that consist of a number of interoperating components distributed across multiple resources. Grid workflows are powerful because

they support the integration of computations, data, scientific instruments and visualization-software while leveraging multiple Grid resources. Grid workflows have been specified for many different scientific domains including physics [31] gravitational wave physics [25], geophysics [40], astronomy [15] and bioinformatics [36]. Accordingly, there have been many projects to date that support Grid workflows, to name a few, *Triana* [47], *Taverna* [37][42], *Kepler* [35], *GrADS* [17] and *P-Grade* [33].

A specialized form of workflow allows the creation of “parameter sweeps”, where a computational model is run repeatedly with different input parameters. Using this approach it is possible to explore different design options, and perform more robust science than previously possible. A number of systems support parameter sweep workflows, including APST [21], the NASA IPG (Information Power Grid) parameter process specification tool [50] and our own Nimrod/G [1][4][5][7].

Apart from these specific environments, programmers can adopt any one of a number of techniques for building distributed applications. These might build on standard languages like Java, and may use special message passing libraries like MPICH/G. Further, tools like Ninf-G [46] and NetSolve [22] provide powerful remote procedure call systems that can invoke arbitrary procedures as well as specific services such as linear algebra. Finally, Web Services provide a generic and standards based mechanism for building large distributed applications, and these underpin the newly developed Web Services Resource Framework (WSRF) [24].

2.2 Deployment

Traditionally, deployment is often combined with development, and may involve little more than copying the executable program to some partition in the file system, and perhaps registering the software with the operating system. The difficulty in deployment is often underestimated because modern installation techniques, such as those used by Microsoft and Apple, appear to simplify the process enormously. However, in a Grid environment, deployment is much more complex because of the need to install the code on a range of heterogeneous machines, geographically distributed and in different administrative domains. In the Grid deploying an application means building and installing the software on a range of different platforms, taking account of issues such as different instruction sets, operating systems, file system structures and software libraries. To date, this phase is often performed manually, which is both error-prone and does not scale to large Grids. For example, in order to deploy an application across 500 computers, a user would typically need to log into each of the 500 sequentially, compiling, linking and installing the software. Our own experiences at deploying a quantum chemistry package over a handful of resources have identified this as a serious bottleneck [43][44].

Surprisingly, there has been little work on the deployment problem, and none of the current middleware projects addresses deployment across heterogeneous resources. Some researchers have suggested solving the problem by taking a system-centric view, as is done in systems that configure a homogeneous compute cluster [18][14][29][30][13]. In this model, an application image is produced for the entire

Grid, and then copied from a central resource. However, this central approach is in conflict with the philosophy of the Grid, which favours a decentralized approach. Moreover, it does not handle significant heterogeneity because each resource could, in the worst case, require a tailored version of the software. One of the few systems that views deployment in a decentralized way is GridAnt [48].

2.3 Testing and Debugging

Testing and debugging software is already challenging on a local workstation. In the Grid this phase is particularly difficult, because the software must be tested and debugged on a range of different platforms, distributed geographically and in different administrative domains. Traditional testing and debugging tools are unable to provide the support required. At present, the only feasible way of debugging a piece of Grid software is for the programmer to log into the remote system and run a conventional debugger such as *gdb* [41]. This technique does not scale to large Grids, and is not practical for workflows that are distributed across multiple resources. Debugging is a serious challenge in the Grid, partly because an application must often execute on a range of different platforms. In this environment, it is not uncommon for a program to work correctly on one machine, but fail in subtle ways when the software is ported or moved to another platform. Traditional debugging techniques usually force the programmer to debug the ported application from scratch, making the task complex and time consuming.

2.4 Execution

This phase typically means scheduling and coordinating execution, using a variety of resources concurrently. Of the phases discussed to date, execution has attracted the most attention. There are many different ways of starting, scheduling and controlling the execution of Grid software, ranging from direct interface to the middleware through to sophisticated scheduling and orchestration systems.

Web Services and Globus [27] provide rudimentary mechanisms for starting jobs on a given remote, and these services can build more complex multi-resource schedules. For example, Kepler contains methods for scheduling and controlling the execution of a workflow and uses the Globus GRAM interface to execute the various workflow actors [12]. Other middleware, such as Condor-G [26][34] and APST [21], use sophisticated scheduling algorithms to enforce quality of service metrics. For example, APST can minimize the total execution time of a workflow. Systems such as Cactus [11] provide techniques for migrating computations across Grid resources, thus the computation adapts to the variability in the resource availability.

3. Grid Middleware

Figure 2 shows a traditional software hierarchy. Here, e-Science applications use services that are exposed by both the platform infrastructure and middleware such as Globus and Unicore [39]. In our experience, whilst powerful, these services are typically too low level for many e-Science applications. As a result, there is a significant ‘semantic gap’ between them, because the application needs are not matched by the underlying middleware services. Moreover, they do not support the software lifecycle thereby making software development difficult and error-prone.

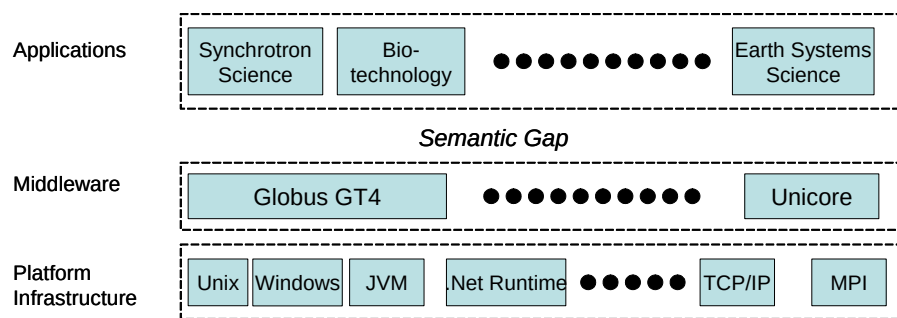


Figure 2 – Traditional software hierarchy

To solve these problems, we propose a new hierarchy as shown in Figure 3. The existing middleware is renamed lower-middleware, and an upper middleware layer is inserted. This upper middleware layer is designed to narrow the semantic gap between existing middleware and applications. Importantly, it hosts a range of interoperating tools that will form the e-Scientists workbench, thus supporting the major phases of the software development lifecycle as well as the applications themselves.

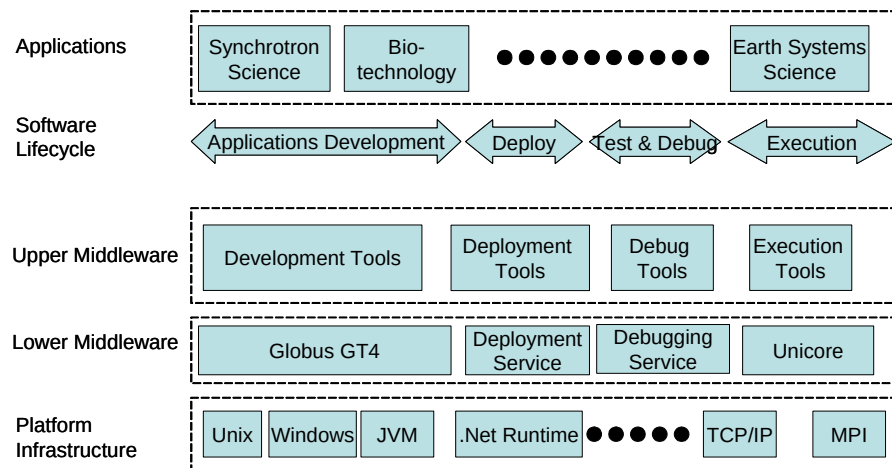


Figure 3 – New software hierarchy

4. Upper Middleware and Tools

Our research group has built a number of software tools that address some of the challenges sited in Section 2, as shown in Figure 4. In particular, Nimrod and GriddleS target software development; Guard focuses on debugging; Grid Work Bench and DistAnt target deployment and Nimrod, GriddLeS, Active Sheets, REMUS and the Nimrod Portal all focus on execution. In this section we provide a very brief overview of these tools.

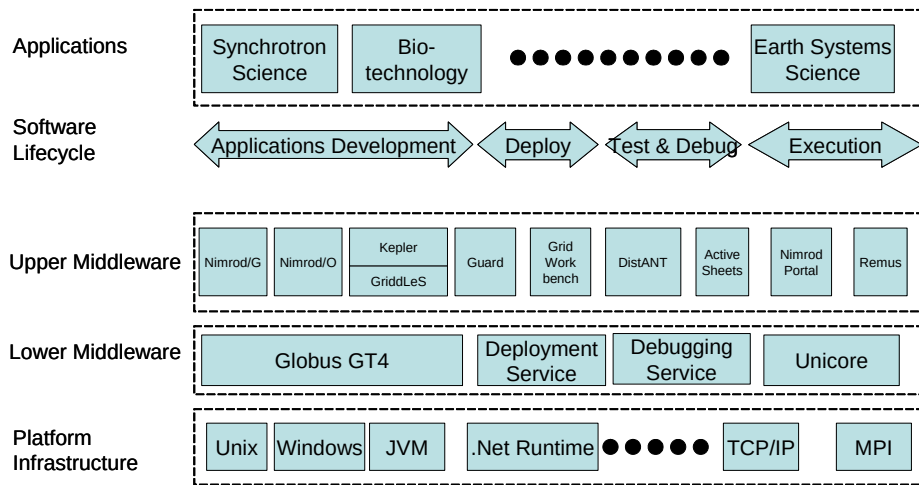


Figure 4 – Monash Grid Tools

4.1 Development

Nimrod/G and GriddLeS [6][8], address some of the challenges in creating of Grid software. Nimrod/G manages the execution of studies with varying parameters across distributed computers. It takes responsibility for the overall management of an experiment as well as the low-level issues of distributing files to remote systems, performing the remote computations, and gathering the results. When users describe an experiment to Nimrod/G, a declarative plan file is developed that describes the parameters, their default values, and the commands needed to perform the work. Apart from this high-level description, users are freed from much of the complexity of the Grid. As a result, Nimrod/G has been very popular among application scientists. Nimrod/O is a variant of Nimrod/G that performs a guided search of the design space rather than exploring all combinations. Nimrod/O allows users to phrase questions such as: "What set of design parameters will minimize (or maximize) the output of my model?" If the model computes metrics such as cost and lifetime, it is then possible to perform automatic optimal design. A commercial version of Nimrod, called EnFuzion, has been produced [16].

GriddLeS, on the other hand, provides a very flexible input-output model that makes it possible to build workflows from legacy applications (written in Fortran, C,

etc) thereby leveraging the enormous amount of scientific software that already exists. GriddLeS allows existing programs to transparently access local and remote files, as well as data that is replicated across multiple servers using Grid middleware such as the Storage Resource Broker [38] and the Globus Replica Location Service [23]. It also allows workflows to pipe data from one application to another without any changes to the underlying data access model. In order to support scientific workflows, we have coupled GriddLeS with the Kepler workflow system. Kepler is an active open source cross-project, cross-institution collaboration to build and evolve a scientific workflow system on top of the Ptolemy II system. Kepler allows scientists from multiple domains to design and execute scientific workflows. It includes two dataflow-based computation models, Process Networks (PN) and Synchronous Data Flow (SDF), and these can be used to define the “orchestration semantics” of a workflow. Simply by changing these models, one can change the scheduling and overall execution semantics of a workflow. By combining Kepler and GriddLeS, a user has significant flexibility in choosing the way data is transferred between the individual components, and this can be done without any changes to the application source.

4.2 Deployment

We are currently developing a few different tools to solve the deployment problem, specifically DistAnt and GWB. DistAnt provides an automated application deployment system with a user-oriented approach [30]. It is targeted at users with reasonable knowledge of the application they are deploying, but strictly limited grid computing knowledge, resource information and importantly, resource authority. DistAnt provides a simple, scalable and secure deployment service and supports a simple procedural deployment description.

DistAnt supports application deployment over heterogeneous grids by virtualizing certain grid resource attributes to provide a common application deployment gateway, deployment description, file system structure and resource description. To manage remaining resource heterogeneity DistAnt supports sub-grids, to divide an unmanageable heterogeneous grid into manageable sets of like resources, categorized by resource attributes that can be queried. Sub-grids provide a framework to execute environment specific remote build routines, compile an application over a set of resource platforms and redistribute binaries to the entire grid. DistAnt also supports definition and deployment of application dependencies. DistAnt enables deployment of a complex native application over an uncharacterized heterogeneous grid, assuming nothing about grid resources.

Furthermore, integration of DistAnt into Nimrod/G, provides an overall environment enabling grid scale application development, deployment and execution.

In addition to DistAnt, we are building a rich interactive development environment (IDE), called Grid Work Bench (GWB). GWB is based on the public domain platform Eclipse [32], and supports the creation, management, distribution and debugging of Grid applications. GWB provides specific functionality to help programmers manage the complexity and heterogeneity of the Grid.

4.3 Testing and Debugging

The Guard debugger targets the process of testing and debugging in the Grid [2][3]. Specifically, it solves some of the problems discussed in Section 2.3 concerning programs that fail when they are ported from one Grid resource to another. We use a new methodology called relative debugging, which allows users to compare data between two programs being executed. Relative Debugging is effectively a hybrid test-and-debug methodology. While traditional debuggers force the programmer to understand the expected state and internal operation of a program, relative debugging makes it possible to trace errors by comparing the contents of data structures between programs at run time. In this way, programmers are less concerned with the actual state of the program. They are more concerned with finding when, and where, differences occur between the old and new code. The methodology requires users to begin by observing that two programs generate different results. They then move back iteratively through the data flow of the codes, to determine the point at which different answers appear. Guard supports the execution of both sequential and parallel programs on a range of platforms. It also exists for a number of different development environments. Because Guard uses a client-server architecture, it is possible to run a debug client on one Grid resource and have it debug an application running on another one, removing the need for users to log into the target system. Further, Guard uses a platform neutral data representation called AIF [49] which means the client and debug servers can run on different types of architecture.

We are concurrently developing a WSRF compliant debug service that will allow high level tools like the GRB to debug applications across multiple Grid resources. This debug service will interoperate with the Globus GRAM interface, thus jobs launched by the GRAM can be debugged in a secure and efficient way using the additional interface.

4.4 Execution

Nimrod provides significant support during the execution of parameter sweeps, including a sophisticated scheduler that enforces real time deadlines. This economy allows users to specify soft real time deadlines that are enforced by trading units in a computational economy [19][20]. Using this approach the system can provide a quality of service that is proportional to the amount of currency a user wishes to expend on an experiment – in short, the more a user pays, the more likely they are to meet their deadline at the expense of another user. The Nimrod scheduler supports two types of inter-task constraints, namely parallel and sequential dependencies. Parallel tasks are executed concurrently providing there are sufficient computational resources. Typically, these tasks pertain to different parameter values in a parameter sweep and can be executed in parallel. However, it is possible to specify special sequential parameters (called seqameters, as opposed to parameters) that force the order of the execution to be sequential. This means that one task may be dependent on the output from another, and its execution can be stalled until the data is available.

The Nimrod Portal and Active Sheets address the execution phase of the life cycle. The Nimrod Portal allows users to create Nimrod experiments from a web interface.

It supports the creation of the plan files discussed above using a graphical user interface, the management of the test bed (and associated Globus issues such as certificate management), and control of the experiment as it executes. Active Sheets [10] allows users to set up and execute an experiment from a familiar spreadsheet interface. Individual cells can invoke Nimrod/G to perform one simulation run; multiple data independent cells can be used to specify an entire “what if” experiment. Because the system is embedded in Microsoft Excel, all normal data manipulation and charting tools are available for post analysis (a feature that is popular with users).

REMUS is an execution environment that helps users build complex Grid applications across firewalls and different administrative domains [45]. REMUS provides mechanisms that reroute traffic through approved channels without compromising the security of any site. It effectively handles heterogeneity in security mechanisms, allowing applications to communicate when there is no common security framework.

5. Conclusion

In this paper we have provided a very brief overview of the challenges in building software for the Grid. We have focused on four phases of a software lifecycle, namely development, deployment, testing and debugging. We have shown that it is possible to split Grid middleware into two layers – one that addresses low level issues and a higher application focused layer. This latter layer can support software tools that make the software development task easier. We have discussed a number of tools developed by the author that simplify the software development task.

Acknowledgments

The author wishes to acknowledge a number of people who have contributed to the work described in this document, including Shahaan Ayyub, Rajkumar Buyya, Phillip Chan, Clement Chu, Colin Enticott, Jagan Kommineni, Donny Kurniawan, Slavisa Garic, Jon Giddy, Wojtek Goscinski, Tim Ho, Andrew Lewis, Tom Peachey, Jeff Tan and Greg Watson.

The projects are supported by a variety of funding agencies, including the Australian Research Council, the Australian Department of Communications, Arts and Information Technology (DCITA), the Australian Department of Education, Science and Technology (DEST), Microsoft, IBM and Hewlett Packard.

References

- 1 Abramson D, Lewis A, Peachey T, Fletcher, C., “An Automatic Design Optimization Tool and its Application to Computational Fluid Dynamics”, SuperComputing 2001, Denver, Nov 2001.

- 2 Abramson D., Foster, I., Michalakes, J. and Susic R., "Relative Debugging: A new paradigm for debugging scientific applications", *Communications of the Association for Computing Machinery (CACM)*, Vol. 39, No 11, pp 67 - 77, Nov 1996.
- 3 Abramson D., Foster, I., Michalakes, J. and Susic R., "Relative Debugging and its Application to the Development of Large Numerical Models", *Proceedings of IEEE Supercomputing 1995*, San Diego, December 95. Paper on CD, no pages
- 4 Abramson D., Susic R., Giddy J. and Hall B., "Nimrod: A Tool for Performing Parametrised Simulations using Distributed Workstations", *The 4th IEEE Symposium on High Performance Distributed Computing*, Virginia, August 1995.
- 5 Abramson, D., Lewis, A. and Peachy, T., "Nimrod/O: A Tool for Automatic Design Optimization", *The 4th International Conference on Algorithms & Architectures for Parallel Processing (ICA3PP 2000)*, Hong Kong, 11 - 13 December 2000.
- 6 Abramson, D. and Kommineni, J., "A Flexible IO Scheme for Grid Workflows". *IPDPS-04*, Santa Fe, New Mexico, April 2004.
- 7 Abramson, D., Giddy, J. and Kotler, L. High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?, *International Parallel and Distributed Processing Symposium (IPDPS)*, pp 520- 528, Cancun, Mexico, May 2000.
- 8 Abramson, D., Kommineni, J. and Altintas, I. "Flexible IO services in the Kepler Grid Workflow Tool", to appear, *IEEE Conference on e-Science and Grid Computing*, Melbourne, Dec 2005.
- 9 Abramson, D., Roe, P., Kotler L and Mather, D., "ActiveSheets: Super-Computing with Spreadsheets". *2001 High Performance Computing Symposium (HPC'01)*, *Advanced Simulation Technologies Conference*, April 22-26, 2001, pp 110 – 115, Seattle, Washington (USA).
- 10 Abramson, D., Roe, P., Kotler L and Mather, D., "ActiveSheets: Super-Computing with Spreadsheets". *2001 High Performance Computing Symposium (HPC'01)*, *Advanced Simulation Technologies Conference*, April 22-26, 2001, pp 110 – 115, Seattle, Washington (USA).
- 11 Allen G. and Seidel, E. "Collaborative Science: Astrophysics Requirements and Experiences", in *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*, Ed: Ian foster and Carl Kesselmann, p. 201-213, (2004)
- 12 Altintas, I. Berkley, C. Jaeger, E. Jones, M. Ludäscher B. and 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.
- 13 Anderson P. and Scobie. A. "LCFG: The Next Generation", *UKUUG Winter Conference*, 2002.
- 14 Anderson, P. Goldsack, P. Paterson, J. "SmartFrog meets LCFG - Autonomous Reconfiguration with Central Policy Control", *2002 Large Installations Systems Administration Conference*, 2003
- 15 Annis, J., Zhao, Y. et al., "Applying Chimera Virtual Data Concepts to Cluster Finding in the Sloan Sky Survey," *Technical Report GriPhyN-2002-05*, 2002.
- 16 Axceleon Inc, <http://www.axceleon.com>
- 17 Berman et al, "The GrADS Project: Software Support for High-Level Grid Application Development", *International Journal of High Performance Computing Applications*, Winter 2001 (Volume 15, Number 4), pp. 327-344.
- 18 Bruno, G. Papadopoulos P. and Katz., M. "Npaci rocks: Tools and techniques for easily deploying manageable linux clusters". *Cluster 2001*, 2001
- 19 Buyya, R., Abramson, D. and Giddy, J. "Nimrod/G: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid", *HPC Asia 2000*, May 14-17, 2000, pp 283 – 289, Beijing, China

- 20 Buyya, R., Abramson, D. and Venugopal, S. "The Grid Economy", Special Issue on Grid Computing, Proceedings of the IEEE, Manish Parashar and Craig Lee (editors), Volume 93, Issue 3, 698-714pp, IEEE Press, New York, USA, March 2005.
- 21 Casanova H. and Berman, F. "Parameter Sweeps on The Grid With APST", chapter 26. Wiley Publisher, Inc., 2002. F. Berman, G. Fox, and T. Hey, editors.
- 22 Casanova H. and Dongarra, J., "Netsolve: A Network Server for Solving Computational Science Problems," Supercomputing Applications and High Performance Computing, vol. 11, no. 3, pp. 212-223, 1997.
- 23 Chervenak, A.L. Palavalli, N. Bharathi, S. Kesselman C. and Schwartzkopf, R. "Performance and Scalability of a Replica Location Service," Proceedings of the International IEEE Symposium on High Performance Distributed Computing (HPDC-13), June 2004
- 24 Czajkowski, K. Foster, I. Frey, J. et al. "The WS-Resource Framework, Version 1.0", 03/05/2004, <http://www.globus.org/wsrf/>
- 25 Deelman, E., Blackburn, K. et al., "GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists," 11th Intl Symposium on High Performance Distributed Computing, 2002.
- 26 Douglas Thain, Todd Tannenbaum, and Miron Livny, "Condor and the Grid", in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley, 2003.
- 27 Foster I. and Kesselman C., Globus: A Metacomputing Infrastructure Toolkit, International Journal of Supercomputer Applications, 11(2): 115-128, 1997.
- 28 Foster, I and Kesselman, C. (editors), The Grid: Blueprint for a New Computing Infrastructure, ISBN: 1558609334, 2nd edition, Morgan Kaufmann, USA, November 18, 2003.
- 29 Goldsack, P. Guijarro, J. Lain, A. et al, "SmartFrog: Configuration and Automatic Ignition of Distributed Applications", HP Labs, UK, 2003, <http://www.hpl.hp.com/research/smartfrog/>
- 30 Goscinski, W. Abramson, "D. Distributed Ant: A System to Support Application Deployment in the Grid", Fifth IEEE/ACM International Workshop on Grid Computing, 2004.
- 31 GriPhyN, www.griphyn.org
- 32 <http://www.eclipse.org>
- 33 Kacsuk, P., Cunha, J.C., Dózs, G., Lourenco, J., Antao, T., Fadgyas, T., "GRADE: A Graphical Development and Debugging Environment for Parallel Programs", Parallel Computing Journal, Elsevier, Vol. 22, No. 13, Feb. 1997, pp. 1747-1770
- 34 Litzkow, M., Livny, M. and Mutka, M. Condor - A Hunter of Idle Workstations". Proceedings of the 8th International Conference on Distributed Computing Systems. San Jose, Calif., June, 1988.
- 35 Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger-Frank, E., Jones, M, Lee, E., Tao J. and Zhao, Y. "Scientific Workflow Management and the Kepler System", Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows, 2005.
- 36 NPACI, "Telescience, <https://gridport.npaci.edu/Telescience/>."
- 37 Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M., Wipat, A. and Li., P. "Taverna: A tool for the composition and enactment of bioinformatics workflows", Bioinformatics Journal 20(17) pp 3045-3054, 2004, doi:10.1093/bioinformatics/bth361.
- 38 Rajasekar, A. Wan, M. Moore, R. Schroeder, W. Kremenek, G. Jagatheesan, A. Cowart, C. Zhu, B. Chen S. and Olschanowsky, R. "Storage Resource Broker - Managing Distributed Data in a Grid," Computer Society of India Journal, Special Issue on SAN, Vol. 33, No. 4, 2003, pp. 42-54.

- 39 Romberg, M. "The UNICORE Architecture–Seamless Access to Distributed Resources", Proceedings of the Eight IEEE International Symposium on High Performance Computing, Redondo Beach, CA, USA, August 1999, Pages 287-293.
- 40 Southern California Earthquake Center's Community Modeling. Environment, <http://www.scec.org/cme/>.
- 41 Stallman, R. Debugging with GDB – The GNU Source Level Debugger, Edition 4.12, Free Software Foundation, January 1994.
- 42 Stevens, R. Tipney, H.J. Wroe, C. Oinn, T. Senger, M. Lord, P. Goble, C.A. Brass A. and Tassabehji M. "Exploring Williams-Beuren Syndrome Using myGrid", Proceedings of 12th International Conference on Intelligent Systems in Molecular Biology, 31st Jul-4th Aug 2004, Glasgow, UK, published Bioinformatics Vol. 20 Suppl. 1 2004, pp 303-310.
- 43 Sudholt, W., Baldridge, K., Abramson, D., Enticott, C. and Garic, S. "Parameter Scan of an Effective Group Difference Pseudopotential Using Grid Computing", New Generation Computing 22 (2004) 125-135.
- 44 Sudholt, W., Baldridge, K., Abramson, D., Enticott, C. and Garic, S., "Application of Grid computing to parameter sweeps and optimizations in molecular modeling", Future Generation Computer Systems, 21 (2005), 27-35.
- 45 Tan, J, Abramson, D. and Enticott, C. "Bridging Organizational Network Boundaries on the Grid", IEEE Grid 2005, Seattle, Nov 2005.
- 46 Tanaka, Y. Takemiya, H. Nakada, H. and Sekiguchi, S. "Design, implementation and performance evaluation of GridRPC programming middleware for a large-scale computational grid," Fifth IEEE/ACS International Workshop on Grid Computing, pp. 298–305, 2005.
- 47 Taylor, I., Wang, I., Shields, M. and Majithia, S. "Distributed computing with Triana on the Grid" In Concurrency and Computation:Practice and Experience , 17(1-18), 2005.
- 48 von Laszewski, G. Alunkal, B. Amin, K. Hampton, S and Nijssure, S. GridAnt-Client-side Workflow Management with Ant, 2002, <http://www.unix.globus.org/cog/projects/gridant/>
- 49 Watson, G. and Abramson, D. "The Architecture of a Parallel Relative Debugger", 13th International Conference on Parallel and Distributed Computing Systems - PDCS 2000, August 8 - 10, 2000.
- 50 Yarrow, M. McCann, K. Biswas, R. and Van der Wijngaart, R. "An Advanced User Interface Approach for Complex Parameter Study Process Specification on the Information Power Grid", Proceedings of the 1st Workshop on Grid Computing (GRID 2002), Bangalore, India, Dec. 2000.