A Summary of Grid Computing Environments

Geoffrey Fox[1,2,3], *Dennis Gannon*[2], *Mary Thomas*[4]
[1]Community Grid Computing Laboratory, Indiana University
501 N Morton Suite 224, Bloomington IN 47404
[2]Computer Science Department, Indiana University
[3]School of Informatics and Physics Department, Indiana University
[4]Texas Advanced Computing Center, The University of Texas at Austin, 10100 Burnet Road, Austin, Texas 78758
gcf@indiana.edu,gannon@cs.indiana.edu,mthomas@tacc.utexas.edu

1 Introduction

This short paper summarizes a set of 28 papers gathered together by the GCE (Grid Computing Environment) group of the Global Grid Forum. This set is published in 2002 as a special issue of Concurrency and Computation: Practice and Experience. The papers are listed at the end of this report together with associated papers in a Grid Computing book [32].

We can define a Grid Computing Environment as a set of tools and technologies that allow users "easy" access to Grid resources and applications. Often it takes the form of a Web portal that provides the user interface to a multi-tier Grid application development stack, but it may also be as simple as a Grid Shell that allows a user access to and control over Grid resources in the same way a conventional shell allows the user access to the file system and process space of a regular operating system.

2 Overall Classification

Grid Computing Environments can be classified in several different ways. One straightforward classification is in terms of technologies used. The different projects differ in terms of languages used, nature of treatment of objects (if any), use of particular technology like Java servlets, the Globus toolkit, or GridFTP, and other implementation issues. Some of these issues are important for performance or architecture but often can look to the user as not so important. For instance, there is a trend to use more heavily Java, XML and Web Services but this will only be interesting if the resultant systems have important properties such as better customizability, sustainability and ease of use without sacrificing too much in areas like performance. Maybe the ease of development using modern technologies is shown up as greater functionality in the GCE for a given amount of implementation effort. Technology differences in the projects are important but more interesting at this stage are the differences in capabilities and the model of computing explicit or implicit in the GCE.

All GCE systems assume there are some backend remote resources (the Grid) and endeavor to provide convenient access to their capabilities. This implies one needs some sort of model for "computing". At the simplest this is running a job, which already has non trivial consequences as data usually needs to be properly set up, and access is required to the running job status and final output. More complicatedly, computing requires coordinated gathering of data, many simulations (either linked at a given time or following each other), visualization, analysis of results etc. Some of these actions require substantial work with other researchers and sharing of results and ideas are needed. This leads to the concept of GCE collaboratories supporting sharing among scientific teams working on the same problem area.

We can build a picture of different GCE approaches by viewing the problem as some sort of generalization of the task of computing on a single computer. So we can highlight the following classes of features:

1) Handling of the basic components of a distributed computing system – files, computing and data resources, programs, and accounts. The GCE will typically interface with an environment like that of the Globus project or a batch scheduler like PBS to actually handle the back-end resources. However it will present the user interfaces to handle these resources. This interface can be simple or complex and often constructed hierarchically to reflect tools built in such a fashion. We can follow the lead of UNIX (and Legion [37] in its distributed extension) and define a basic GCEShell providing access to the core distributed computing functions. For example, JXTA [49] also builds Grid-like capabilities with a UNIX shell model. GCEShell would support running and compiling jobs, moving among file systems etc. GCEShell can have a command line or more visually appealing graphical user interface.
2) The 3-tier model, which is typically used for most systems, implies than any given capability (say run a matrix inversion program) can appear at multiple levels. Maybe there is a backend parallel computer running an MPI job; this is front-ended perhaps as a service by some middle-tier component running on a totally different computer, which could even be in a different security domain. One can "interact" with this service at either level; a high performance I/O transfer at the parallel computing level and/or by a slower middle-tier protocol like SOAP at the service level. These two (or more) calls (component interactions) can represent different functions or the middle tier call can be coupled with a high performance mirror; typically the middle tier provides control and the back end "raw data transfer". The resultant rather complicated model is shown in fig.1. We have each component (service) represented in both middle and HPC (raw) tiers. Intra-tier and inter-tier linkage is shown. Ref. [33] has an excellent review of the different programming models for the Grid.

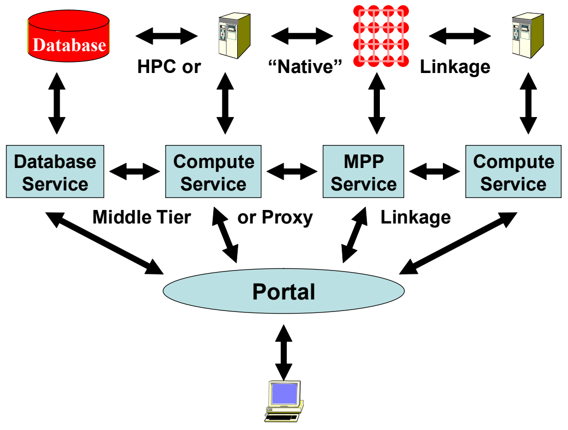

Figure 1. Middle-Tier and Raw (HPC) Linked Components of an Application

3) One broadly important general-purpose feature is Security (authentication, authorization and privacy), which is addressed in some way or other by essentially all environments.
4) Data management is a broadly important topic, which gets even more important on a distributed system than it is on single machines. It includes file manipulation, databases and access to raw signals from instruments such as satellites and accelerators.

5) One augments the basic GCEShell with a library of other general purpose tools and this can be supported by the GCE. Such tools include (Grid)FTP, (Grid)MPI, parameter sweep and more general workflow, and the composition of GCEShell primitives.
6) Other higher-level tools are also important and many tend to be rather application dependent; visualization and intelligent decision support as to what type of algorithm to use can be put here.
7) Looking at commercial portals, one finds that they usually support sophisticated user interfaces with multiple sub-windows aggregated in the user interface. The Apache Jetspeed project is a well-known toolkit supporting this [47]. This user interface aggregation is often supported by a GCE. This aggregation is not stressed in any paper in this special issue although implicitly it is provided.

As well as particular features, a GCE usually implies a particular computing model for the Grid and this model is reflected in the GCE architecture and the view of the Grid presented to the user. For example object models for applications are very popular and this object view is reflected in the view of the Grid presented to the user by the GCE. Note the programming model for a GCE is usually the programming of rather large objects – one can describe programs and hardware resources as objects without this object model necessarily changing the software model used in applications.

With this preamble, we can now classify the papers in this special issue. There are, as always, no absolute classifications for a complex topic like distributed Grid systems. Hence it is often the case that these projects can be looked at from many overlapping points of view.

3 Summary of GCE Projects and Features

3.1 Technology for building GCE Systems

There is one set of papers describing convenient interfaces to the Grid and in particular the Globus toolkit [32, 46]. Refs. [6, 38], [14], [15] and [27] describe respectively Java, CORBA, Python and Perl interfaces to the Globus toolkit. These are the basic building blocks of full GCE's. The middleware of Ref. [1] as well as the problem solving environments in Refs. [7], [8] and [20] build on top of the Java Commodity Grid Kit [6]. The portals described in Ref. [26] build directly on top of the Perl Commodity Grid Kit [27].

Ref. [1] is higher level than the toolkit of Ref. [6] and builds a suite of JavaBeans suitable for Java based GCE environments; the technology is designed to support JSP (Java Server Pages) displays. Ref. [9] provides C support for interfacing to the Globus toolkit and portals exposing the toolkit's capabilities can be built on the infrastructure of this paper.

Ref. [17] proposes interesting XML based technology for supporting the runtime coupling of multidisciplinary applications with matching of geometries.

Ref. [21, 43] notes that current Grid architectures build more and more on message-based middleware and this is particularly clear for Web Services; this paper designs and prototypes a possible event or messaging support for the Grid.

Ref. [28] describes a rather different technology; namely a Grid simulator aimed at testing new scheduling algorithms.

3.2 Largely Problem Solving Environments

We have crudely divided those GCE's offering user interfaces into two classes. One class focusing on a particular application (set) which are sometimes called application portals or Problem Solving Environments (PSE's). The second class offer generic application capabilities and have been termed user portals; in our notation introduced above, we can call them GCEShell portals. Actually one tends to have a hierarchy with PSE's building on GCEShell portals; the latter building on middleware like GPDK [1]; GPDK builds on the Java CoG Kit [6] which itself builds on the Globus toolkit that finally builds on the native capabilities of the Grid component resources. This hierarchy is for one set of technologies and architecture but other approaches are similarly built in a layered fashion.

Several papers in this issue include discussion of Grid PSE's. Ref. [5] has an interesting discussion of the architectural changes to a "legacy" PSE consequent on switching to a Grid Portal approach. Ref. [11] illustrates the richness of PSE with a survey of several operational systems; these share a common heritage with the PSE's of Ref. [16] although the latter paper is mainly focused on a recommender tool described later.

Five further papers describe PSE's that differ in terms of GCE infrastructure used and applications addressed. Ref. [7] describes two PSE's built on top of a GCEShell portal with an object computing model. A similar portal is the XCAT Science portal [29], which is based on the concept of application Notebooks that contain web pages, Python scripts and control code specific to an application. In this case the Python script code plays the role of the GCEShell. The astrophysics collaboratory [20] includes the Globus toolkit link via Java [6] and the portal toolkit of [1]; it also interfaces to the powerful Cactus distributed environment [31]. Ref. [18, 41] presents a portal for computational physics using Web services – especially for data manipulation services. The Polder system [24] and SCIRun [25] offer rich visualization capabilities within several applications including biomedicine. SCIRun has been linked to several Grid technologies including NetSolve [10], and it supports a component model (the CCA [48] not directly covered in this special issue) with powerful workflow capabilities.

3.3 Largely Basic GCEShell Portals

Here we describe the set of portals designed to support generic computing capabilities on the Grid. Ref. [3] is interesting as it is a Grid portal designed to support the stringent requirements of DoE's ASCI program. This reflects not only security and performance issues but the particular and well established computing model for the computational physicists using the ASCI machines. Ref. [4] describes a portal interface to the very sophisticated Legion Grid which has through the Legion Shell a powerful generic interface to the shared object (file) system supported by Legion [37]. This paper also describes how specific problem solving environments can be built on topic of the basic GCEShell portal.

Ref. [1] provides, via the Java CoG Kit [6] interfacing to the Globus toolkit, a set of middleware JavaBeans, which are at the GCEShell abstraction level and support

problem solving environments as we have already discussed for Ref. [20]. Unicore [23, 44] was one of the pioneering full featured GCEShell portals developed originally to support access to a specific set of European supercomputers but recently has been interfaced to the Globus toolkit. Unicore has developed an interesting abstract job object (AJO) with full workflow support.
Refs. [7], [13] and [26] describe well developed GCEShell portals on which application specific PSE's have been built. Ref. [26, 45] builds HotPage, a GCEShell on top of GridPort which is middleware using the Perl Community Grid Kit [27] to access the Globus toolkit.

3.4 Security

Security is discussed in most of the papers of this special issue with the Public Key Infrastructure pioneered by the Globus project being most popular. Kerberos is required by some installations (DoD and DoE for instance in the USA) and Grid Computing Environments developed for them [3] [7] [13] are based on this security model.

3.5 Workflow

Workflow corresponds to composing a complete job from multiple distributed components. This is broadly important and is also a major topic within the commercial Web service community. It is also inherently a part of a GCEShell or PSE, since these systems are compositions of specific sequences of tasks. Several projects have addressed this but currently there is no consensus how workflow should be expressed, although several groups have developed visual user interfaces to define the linkage between components. Workflow is discussed in papers [3], [8], [17], [23] and [25]. The latter integrates Grid workflow with the dataflow paradigm, which is well established in the visualization community. Ref. [17] has stressed the need for powerful runtime to support the coupling of applications and this is implicit in other papers including Ref. [8].

3.6 Data Management

Data intensive applications are expected to be critical on the Grid but support of this is not covered in the papers of this special issue. Interfaces with file systems, databases and data transfer through mechanisms like GridFTP are covered in several papers. This is primarily due to the fact that data management software is still relatively new on the grid.

3.7 GCEShell Tools

In our GCE computing model, one expects a library of tools to be built up that add value to the basic GCEShell capabilities. The previous two subsections describe two tools – workflow and data management of special interest and here we present a broad range of other tools which appeared in several papers in this special issue.

Netbuild [2] supports distributed libraries with automatic configuration of software on the wide variety of target machines on the Grids of growing heterogeneity.

NetSolve [10, 40] pioneered the use of agents to aid the mapping of appropriate Grid resources to client needs. Ref. [16] describes a recommendation system which uses detailed performance information to help users on a PSE, choose the best algorithms to address their problem.

Many projects have noted the importance of "parameter sweep" problems where a given application is run many times with different input parameters. Such problems are very suitable for Grid environments and Ref. [22] describes a particular parameter sweep system Nimrod-G. This paper focuses on a different tool – namely a novel scheduling tool based on an economic model of Grid suppliers and consumers. Ref. [34] describes a another well regarded parameter sweep system APST building on the AppLeS application level scheduling system.

HotPage [26, 45] is well known for pioneering the provision of job status information to portals; such a tool is clearly broadly important.

Finally we should stress visualization as a critical tool for many users and here Refs. [25] and [17] describe this area. There are many other important tools like data-mining which fall into this category.

3.8 GCE Computing Model

In the preamble we suggested that it was interesting to consider the computing model underlying Grid Computing Environments. This refers to the way we think about the world of files, computers, databases and programs exposed through a portal. NetSolve described in [10, 40] together with the Ninf effort [30, 35] in Japan has developed the important Network Service model for distributed computing. Rather than each user downloading a library to solve some part of their problem, this task is dispatched to a Network resource providing this computational service. Using Web Services allows one to view NetSolve as a distributed object model for the Grid. Both Ninf and NetSolve support the new GridRPC remote procedure call standard, which encapsulates a key core part of their Grid computing model [33]. GridRPC supports scientific data structures as well as Grid specific security and resource management.

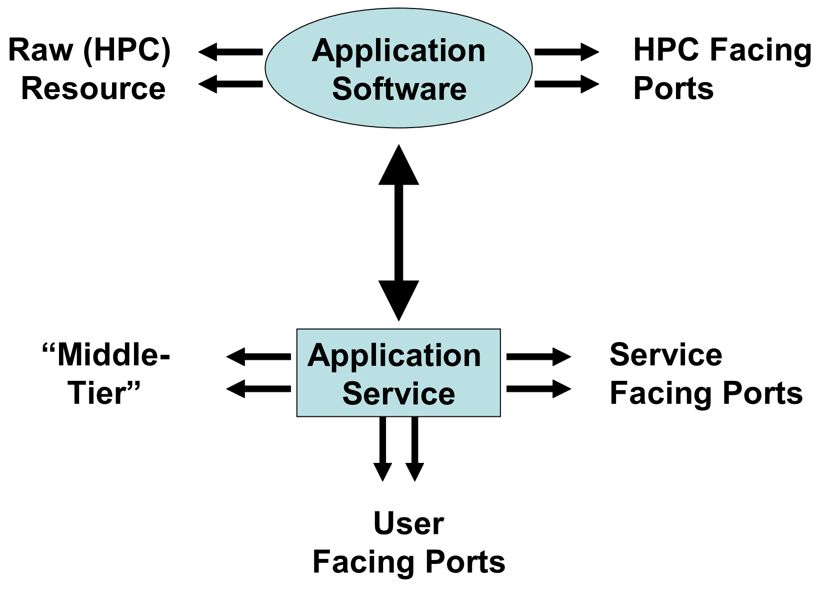

Figure 2. A Proxy Service Programming Model showing 4 types of Interactions :to and from users (portal interface), between proxy and raw resource, other middle-tier components and between other raw (hpc) resources

Ref. [12] describes Grid implementations of MPI (message passing standard for parallel computing), which address the incompatibilities between MPI implementations and binary representations on different parallel computers. Note that in the notation of

Fig. 1, MPI is at the "HPC backend linkage" layer and not at the middleware layer. Ref. [20] supports the Cactus environment [31, 36] which has well developed support for Grid computing at the HPC layer i.e. it supports backend programming interfaces and not the middle-tier GCEShell environment. The astrophysics problem solving environment of Ref. [20] augments Cactus with a full middle tier environment.

Legion in Refs. [4] and [37] built a very complete Grid object model. Ref. [8] describes a CORBA distributed object model for the Grid and Ref. [19, 42] describes the surprisingly hard issues involved in providing interoperability between multiple CORBA GCE's. We can hope that Web services will prove to be easy to make interoperable, as the technology used (XML, SOAP) is more open than CORBA, which has evolved with several often incompatible implementations as listed in Ref. [14].

Refs. [7, 39, [13, 39],[23, 44] and the XCAT Science Portal [29] also present an object model for GCE computing but with one critical feature – namely the middle tier objects are always proxies, which hold the meta-data that describe "real resources" which operate in conventional environments. This proxy strategy appears useful for many Grid resources although the true Network service model of NetSolve is also essential. Let us give a simple example from UNIX and suppose one wanted to send data between two programs (in different machines). One could choose the mechanism within the program and use a simple socket or FTP or RMI interaction mechanism. Alternatively the programs could be written generically with output and input or "standard I/O". The programs could then have the output of one "piped" to the input of the other from a UNIX shell command. Such a hybrid programming model with actions partly specified internally and partly specified at service level is important of the success of the Grid and should be built into programming models for it.

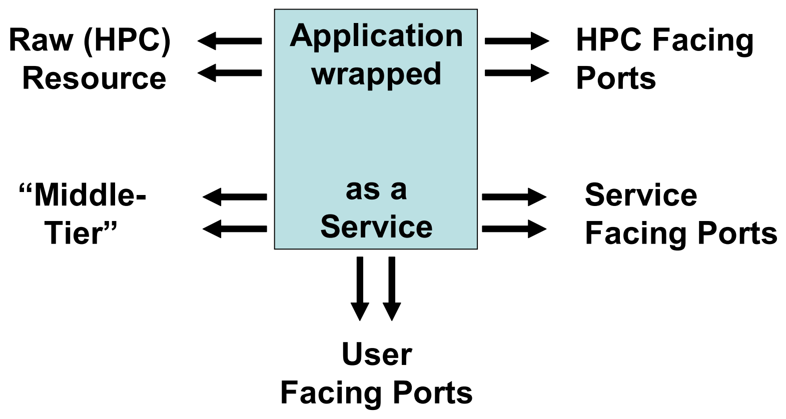

Figure 3. A Wrapped Application Programming Model showing 3 types of Interactions to and from users (portal interface) to and from other middle-tier components and between other raw (HPC) resources

Any GCE computing model should support both the meta-data only and wrapped styles of Grid objects. Actually going back to point 2) in Section 2, the proxy and NetSolve model are not really different as indicated in figures 2 and 3. Both models effectively wrap application (software) resources as objects. In the proxy model, one exposes the interaction between middle-tier and back-end. In the wrapped service model of NetSolve and Ninf, one presents a single entity to the user. In both cases, one can have separate middle-tier and HPC ("real") communication. To complicate the classification, there can of course be a difference between programming model abstraction (proxy or not) and implementation. In the XCAT model, a software component system [48] is used which implements the wrapped service or proxy model. The component system is based on Web Service standards so it is possible that the wrapped service components may be arbitrary Grid Services.

4 References

The first 28 references are the papers in this special issue. We indicate where these are extended by a further article in a book on Grid Computing [32].

1) Jason Novotny, "The Grid Portal Development Kit", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. This is reprinted in Ref. [32].
2) Keith Moore and Jack Dongarra, "NetBuild: Transparent Cross-Platform Access to Computational Software Libraries", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002.
3) Randal Rheinheimer, Steven L. Humphries, Hugh P. Bivens and Judy I. Beiriger, "The ASCI Computational Grid: Initial Deployment", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002.
4) Anand Natrajan, Anh Nguyen-Tuong, Marty A. Humphrey and Andrew S. Grimshaw, "The Legion Grid Portal", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [37].
5) Karen Schuchardt, Brett Didier and Gary Black **, "**Ecce - A Problem Solving Environment's Evolution Toward Grid Services and a Web Architecture", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002.
6) Gregor von Laszewski, Jarek Gawor, Peter Lane, Nell Rehn, and Mike Russell "Features of the Java Commodity Grid Kit", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [38].
7) Tomasz Haupt, Purushotham Bangalore and Gregory Henley, "Mississippi Computational Web Portal", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [39].
8) Andreas Schreiber, "The Integrated Simulation Environment TENT", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002.
9) Giovanni Aloisio and Massimo Cafaro, "Web-based access to the Grid using the Grid Resource Broker Portal", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002.
10) D. Arnold, H. Casanova, and J. Dongarra, "Innovations of the NetSolve Grid Computing System", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [40].

11) Naren Ramakrishnan, Layne T. Watson, Dennis G. Kafura, Calvin J. Ribbens, and Clifford A. Shaffer, "Programming Environments for Multidisciplinary Grid Communities", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002.
12) M. Mueller , E. Gabriel and M. Resch, "A Software Development Environment for Grid Computing", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002.
13) Marlon. E. Pierce, Choonhan Youn, and Geoffrey C. Fox, "The Gateway Computational Web Portal", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [39].
14) Gregor von Laszewski, Manish Parashar, Snigdha Verma, Jarek Gawor, Kate Keahey, and Nell Rehn, "A CORBA Commodity Grid Kit", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [38].
15) Keith Jackson "pyGlobus: A Python interface to the Globus Toolkit", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [38].
16) E. Houstis, A. C. Catlin, N. Dhanjani and J. R. Rice, N. Ramakrishnan and V. Verykios, "MyPYTHIA: A Recommendation Portal for Scientific Software and Services", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002.
17) Jerry A. Clarke and Raju R. Namburu, "A Distributed Computing Environment for Interdisciplinary Applications", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002.
18) William A. Watson III , Ian Bird, Jie Chen, Bryan Hess, Andy Kowalski and Ying Chen, "A Web Services Data Analysis Grid", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [41].
19) Vijay Mann and Manish Parashar, "Engineering an Interoperable Computational Collaboratory on the Grid", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [42].
20) Gregor von Laszewski, Michael Russell, Ian Foster, John Shalf, Gabrielle Allen, Greg Daues, Jason Novotny and Edward Seidel, "Community Software Development with the Astrophysics Simulation Collaboratory", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [36].
21) Geoffrey Fox and Shrideep Pallickara, "An Event Service to Support Grid Computational Environments", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [43].
22) Rajkumar Buyya, David Abramson, Jonathan Giddy, and Heinz Stockinger, "Economics Paradigm for Resource Management and Scheduling in Grid Computing", Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002.

23) Dietmar W. Erwin, “UNICORE – A Grid Computing Environment”, Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [44].
24) K. A. Iskra,R. G. Belleman, G. D. van Albada, J. Santoso, P. M. A. Sloot, H. E. Bal, H. J. W. Spoelder and M. Bubak, “The Polder Computing Environment: a system for interactive distributed simulation”, Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002.
25) Chris Johnson , Steve Parker and David Weinstein, “Component-Based Problem Solving Environments for Large-Scale Scientific Computing”, Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002.
26) M. Dahan, K. Mueller, S. Mock, C. Mills, M. Thomas, “Application Portals: Practice and Experience”, Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [45].
27) S. Mock, M. Dahan, M. Thomas and G. von Lazewski, “The Perl Commodity Grid Toolkit”, Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002. See Ref. [38].
28) Manzur Murshed, Rajkumar Buyya, and David Abramson, “GridSim: A Toolkit for the Modeling and Simulation of distributed resource management and scheduling for Grid Computing”, Concurrency and Computation: Practice and Experience Vol. 14, Grid Computing environments Special Issue 13-14, 2002.
29) ``The XCAT Science Portal,'' S. Krishnan, R. Bramley, M. Govindaraju, R. Indurkar, A. Slominski, D. Gannon, J. Alameda and D. Alkaire, Proceedings SC2001, Nov. 2001, Denver.
30) Ninf network server project http://ninf.apgrid.org/
31) Cactus Grid Computational Toolkit http://www.cactuscode.org
32) Fran Berman, Geoffrey Fox and Tony Hey, ‘Grid Computing: Making the Global Infrastructure a Reality’, John Wiley & Sons Ltd, Chichester, 2003. See http://www.grid2002.org
33) Craig Lee and Domenico Talia, ”Grid Programming Models: Current Tools, Issues and Directions”, Chapter in Ref. [32].
34) Henri Casanova and Fran Berman, “Parameter Sweeps on the Grid with APST”, Chapter in Ref. [32].
35) Hidemoto Nakada, Yoshio Tanaka, Satoshi Matsuoka and Staoshi Sekiguchi, “Ninf-G: a GridRPC system on the Globus Toolkit”, Chapter in Ref. [32].
36) Gabrielle Allen, Tom Goodale, Michael Russell, Edward Seidel and John Shalf, “Classifying and enabling grid applications”, Chapter in Ref. [32].
37) Andrew S. Grimshaw, Anand Natrajan, Marty A. Humphrey, Michael J. Lewis, Anh Nguyen-Tuong, John F. Karpovich, Mark M. Morgan and Adam J. Ferrari, “From Legion to Avaki: The Persistence of Vision”, Chapter in Ref. [32].
38) Gregor von Laszewski, Jarek Gawor, Sriram Krishnan and Keith Jackson, “Commodity Grid Kits - Middleware for Building Grid Computing Environments”, Chapter in Ref. [32].
39) Tomasz Haupt and Marlon E. Pierce, “Distributed object-based grid computing environments”, Chapter in Ref. [32].

40) Sudesh Agrawal, Jack Dongarra, Keith Seymour, and Sathish Vadhiyar, "NetSolve: Past, Present, and Future; A Look at a Grid Enabled Server", Chapter in Ref. [32].
41) William A. Watson, Ying Chen, Jie Chen and Walt Akers, "Storage Manager and File Transfer Web Services", Chapter in Ref. [32].
42) V. Mann and M. Parashar, "DISCOVER: A Computational Collaboratory for Interactive Grid Applications", Chapter in Ref. [32].
43) Geoffrey Fox and Shrideep Pallickara, "NaradaBrokering: An Event Based Infrastructure for Building Scaleable Durable Peer-to-Peer Grids", Chapter in Ref. [32].
44) David Snelling, "Unicore and the Open Grid Services Architecture", Chapter in Ref. [32].
45) Mary Thomas and Jay Boisseau, "Building Grid Computing Portals: The NPACI Grid Portal Toolkit", Chapter in Ref. [32].
46) The Globus Grid Project http://www.globus.org
47) Apache Jetspeed Portal http://jakarta.apache.org/jetspeed/site/index.html
48) Common Component Architecture http://www.cca-forum.org/
49) JXTA Peer-to-Peer Environment http://www.jxta.org