Virtual Experiment Services

Li-ping Shen, Rui-min Shen, and Ming-lu Li

Department of Computer Science & Engineering, Shanghai Jiaotong Univ.,
HuaShan Rd. 1954#, Shanghai, 200030, China
{lpshen, rmshen, mlli}@mail.sjtu.edu.cn

Abstract. There is increasing recognition of the need for laboratory experience that is through these experiences that students could deepen their understanding of the conceptual material, especially for the science and engineering courses. Virtual Experiment has advantages over physical laboratory at many aspects. Nowadays virtual experiments are mostly stand-alone applications without standard interface, which are difficult to reuse. Moreover it is a challenging for compute and data intensive virtual instruments to be reasonably responsive. In this paper, we propose a virtual experiment model based on novel grid service technology. In this model we employ two-layered virtual experiment services to provide cheap and efficient distributed virtual experiment solution. This model could reuse not only virtual instruments but also compositive virtual experiments. In order to reuse successfully-deployed virtual experiment, we advance a uniform schema to describe a virtual experiment plan and process.

1 Introduction

As modern society steps into the information age, e-Learning has taken on increased importance in many facets of life. Universities give more and more courses through web-based courseware, and more and more employees begin to enrich their knowledge through courses provided by company intranet or education institutes. The present web-based courses are always composed of video, audio, figures, exercises and text, with little interactive and personal experience. There is increasing recognition of the need for laboratory experience. It is through these experiences that students could deepen their understanding of the conceptual material, especially for the science and engineering courses. Virtual Experiment (VE) could supply such a gap.

VE is powerful application software system which simulate physical lab environment. With the up-to-date computer and multimedia technology, VE could provide students highly immersion and rich experience. It is a high-valued teaching and learning tool, not only for e-Learning courses but also for traditional courses. It has many advantages over physical laboratory.

- A cost effective way to leverage expensive equipments and maintain physical laboratory by lab attendants. In the face of rapid technology advances, maintaining an up-to-date laboratory presents a significant challenge to universities [14].

R.W.H. Lau et al. (Eds.): ICWL 2005, LNCS 3583, pp. 99 – 110, 2005.

- Provide concurrent on-line instruction, visualization, repeated practice and feedback; break the geographical, lab space and time constraints.
- Provide experiments that can't really be done in the physical lab, e.g. an experimental study of Newton's second law, simulation of a nuclear power plant.
- Flexibility and adaptability. You can adapt a virtual instrument to your particular needs without having to replace the entire devices; however, the users generally cannot extend or customize physical instruments.
- Enabling convenient and economic access to expensive and specialized instruments reuse through remote control, enabling cooperative experiment and research.

Early players of VE include Virtual Physics Laboratory [16] in University of Oregon, Control the Nuclear Power Plant [17] in Swedish Linkopings University, The Interactive Frog Dissection [18] in University of Virginia, Visual Systems Laboratory [19] in University of Central Florida and Oorange for Experimental Mathematics [20] in Technique University Berlin. The common points of these VE are:

- Implemented as stand-alone applications using Java Applet and Virtual Reality techniques.
- Extensive effort have been put into these virtual experiments
- The main components of VE, virtual instruments (VI), are difficult to reuse.
- Technology used is typically beyond an average educator.

It is energy and capital consuming to design and develop a VE/VI to perform the functions of a traditional experiment/instrument. It is a waste of time and energy if we couldn't reuse these VEs/VIs. Not only the simulation VI needs to be reused, but also the expensive physical instruments do. Thanks to the advanced network technologies, scientists now could access remote instruments efficiently [15]. Though the core function is provided by a physical instrument, the interface to the user is the soft panel on the client PC and data are communicated between soft panel and the instrument driver through network which is transparent to the user, so the remote-control enabled instrument is also a VI. It is the trend to provide reusable and efficient VIs for the quick and cheap VE deployment. Albert Ip and Ric Canale introduces a conceptual model of reusable "virtual apparatus" for designing virtual experiments with emphasis on minimizing the technical burden on the teacher by using generic programmable objects in [2]. In their model, Virtual apparatus are software components that can be dynamically combined together to create a virtual experiment. Virtual apparatus could either run on a remote server, or download to the local client, according to the requirement of communication and computation.

In order to make VIs reusable and interoperable, VXIbus Consortium[7] has take great efforts to establish standards to ensure instrument hardware interoperability, while VXI plug&play Systems Alliance[8] addresses the software level interoperability of VIs. For example the VXI-11.1[9] defines TCP/IP-VXIbus Interface Specification and the VPP-4.1 specification [10] provides a standard Virtual Instrument Software Architecture (VISA). But VXI and VXIplug&play standards don't address the implementation of a VE as a whole, i.e. how to organize the instruments into a VE.

They only provide attributes access without functional and taxonomic description of a VI. It is an urgent requirement for us to devise a standard interface of VI and an intelligent mechanism for teachers to design a VE without much unnecessary effort.

It is also a challenge for compute and data intensive VI to be reasonably responsive. The XPort project [15] exploits a combination of advanced Grid services and remote instrument technologies to achieve interactive "better-than-being-there" capabilities for remote experiment planning, operation, data acquisition and analysis with several X-Ray crystallography facilities.

The outline of this paper is as follows. Section 2 describes the design requirements of VE. Section 3 set forth the layered structure of VE Services, which base on the grid services and Globus Toolkit 3. The model of the VE grid employing VE services is introduced at section 4 and section 5 concludes this paper.

2 Design Envisioning of Virtual Experiment

There is no doubt that VE is a complex system, it is an integral running environment which provide the container for involved VIs. On the basis of top-down analysis,VE consists of three parts: programming by programmers, designing by teachers and experimenting by students. A successful VE system must ensure that programmers could develop VE software according to standard architecture and interface for the sake of interoperability and easy maintenance, teachers with no other expertise but their own instructional field could design VE easily, and the students could enjoy and immerse VE anytime and anywhere with graphics interface. So different people may have different responsibilities here in the VE, a teacher need not be a programmer at the same time.

The VE model is based on the component model in software engineering. VE software components can be dynamically combined together to create a VE. There are three major components in the model [2]:

- Virtual Instruments
- Virtual Experiment work bench and
- Virtual Experiment constraints computing

VI is the main component that could be manipulated easily, interoperable, could be assembled together to form new experiment and could reflect behaviors of the real world. The VE work bench is the components container, managing and linking the VIs together. Work bench is the VI communication broker, only through which VIs could interact. The VE constraints computing component denotes the experiment principles, the expressions holding in the VI together. For example, when two moving objects collide on a smooth land (the friction coefficient is zero), then the applicable constraints are the momentum and energy conversation laws as equation 1. When the students interact with the VE by clicking, dragging and so on, these interactions fire up events of the VIs. The events are parsed and processed by the work bench, invoking constraints computing when necessary, altering the parameters of the VI, and creating response to the learner accordingly. In order to be reusable and interoperable, VI must have standard description and interface, and VE must have standard description of the constraints and rules.

m1v1(t1)+m2v2(t1)=m1v1(t2)+m2v2(t2) and 1/2*m1(v1(t1))2+1/2*m2(v2(t1))2=1/2*m1(v1(t2))2+1/2*m2(v2(t2))2 let: mi : the mass of the ith object, i=1,2 vi(tj) : the velocity of the ith object at time tj , i=1,2, j=1,2	(1)

A VE isn't limited or confined to a stand-alone PC. In fact, with recent developments in network technologies and the Internet, it is more common for experiments to use the power of the connectivity for the purpose of task sharing. Typical examples include distributed instruments and monitoring, device remote access, as well as data analysis or result visualization from multiple locations. Most importantly, a successful VE should be reasonably responsive.

3 Virtual Experiment Services

Upon the analysis above, VE firstly is made up of distributed multi-vendor components, which reside in heterogeneous machine within different control domains, and which must provide standard interfaces and descriptions in order for reusability. Secondly, a VE, which must hold all its components together to produce high efficiency, should use open and standard protocols and interfaces. And finally VE demand high QOS and performance to construct a real-time and interactive environment. According to the three point checklist for the gird [5], it is reasonable that we use the novel Grid technology to construct a VE Grid to provide dependable, consistent, pervasive and inexpensive access to VEs.

Our proposed VE architecture is based on the widely acknowledged middleware product, the newly released version of Globus (GT3), which includes an Open Grid Services Architecture [1] implementation to provide an interoperable, industry-usable platform. There are three lays in GT3 architecture [6], from bottom up including: the GT3 core which implements all OGSI specified interfaces and the Grid Security Infrastructure, the GT3 base services which implement both existing Globus Toolkit capabilities(for example The Monitoring and Discovery Service (MDS), Globus Resource Allocation Manager (GRAM), GridFTP, Reliable File Transfer (RFT), Replica Location Service (RLS)) and new capabilities such as reservation and monitoring, and higher-level services.

Fig.1 describes the layered architecture of VE Services. The VE Services are organized in two hierarchical levels: the core VE Services layer and the high-level VE Services layer. The core VE Services layer offers basic services for VE resource lookup and location, and data management. These services include VE directory services, resource allocation services and data management services, which are implemented directly on top of generic grid services. The high-level VE services layer provides services for users to organize and access resources. It consists of VI access services, tool access services and result presentation services.

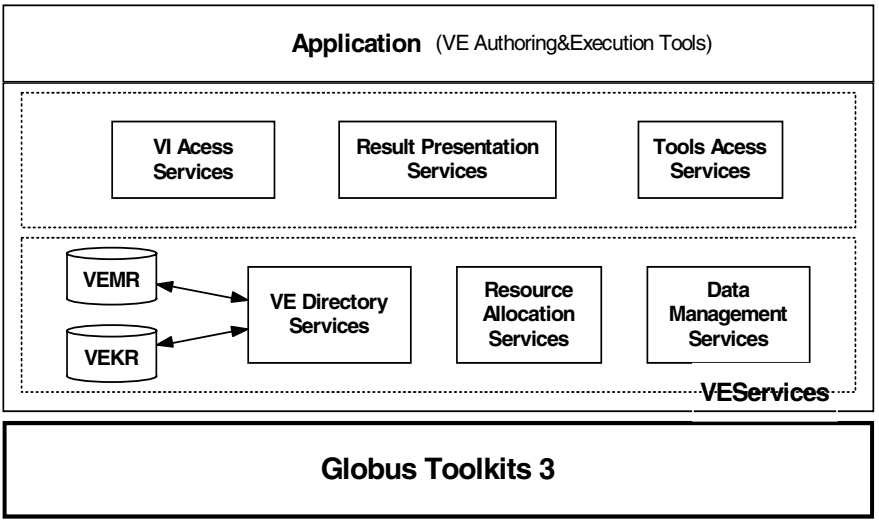

Fig. 1. Layered Architecture of VE Services

3.1 Core VE Services Layer

This layer employs basic grid services to provide data management and resource management. For the VE Services, data are the input/output data for the VI, analysis tools and constraint computing, while resources include VI, analysis & visualization tools and constraints computing tools besides the generic grid resources such as CPU, memory and database. The Core VE Services layer comprises three main services.

VE Directory Service (VEDS)
VEDS extends the basic Globus MDS service and it is responsible for maintaining a description of all resources used in the VE grid and responding to queries of available resources. The resources may be VI Services, tools and algorithms to analyze and visualize data, data source and data sinks, stored VE processes etc. Each metadata instance includes the following information: factory that allows a client to retrieve a reference to the service, category, keywords, the input/output parameters, typical execution time, constraints such as platform and human-readable description. The metadata information is presented by XML documents and is stored in a VE Metadata Repository (VEMR).

Another important repository is the VE Knowledge Repository (VEKR). VEKR stores and provides access to VE process performed within the VE Grid. It warehouses the VE's process information (past experience) and allows this knowledge to be reused. Once users have constructed successful VE processes they wish to be re-used, they can publish them as new services. In order to enable this function, firstly we need uniform description of a VE. The information needed here include the organization of the resources, the resources metadata description, the steps of the process and the experiment principles (constraints).

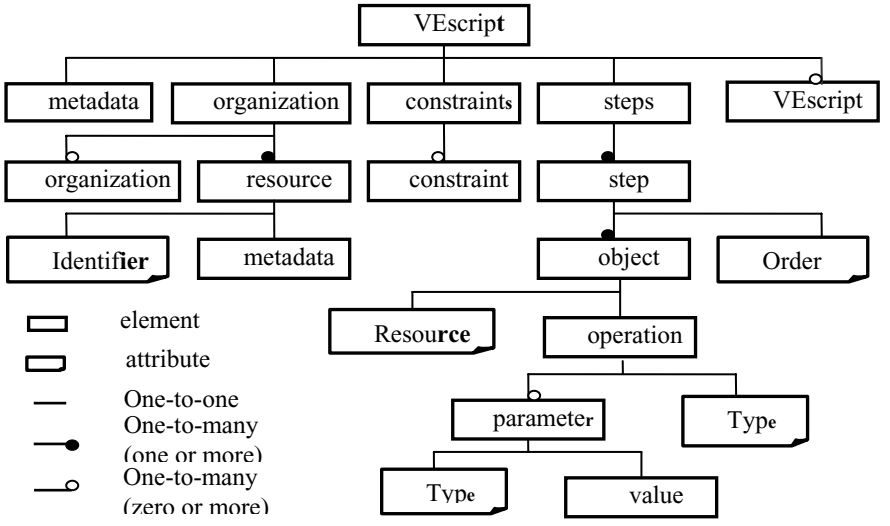

Fig. 2. VE Description Schema

Fig. 2 is our proposed model of the VE description. Vescript is the root element, consisting of four sub-elements: metadata, organization, constraints and steps. Vescript element can have zero or more nested sub-vescripts. Metadata element describes the VE process, including the same information as VEMR metadata. Organization element describes all the resources used during the whole VE process. It includes one or more resource sub-element and zero or more nested sub-organization. The metadata of resource element is extracted from the VEMR. Resource element has an attribute identifier which is unique within the vescript. The constraints element may have zero or more constraint sub-element which is expressed in MathML[21]. The steps element has an order attribute to indicate the operation sequence and one or more object sub-elements. The resource attribute of object references one of the resource identifiers defined in the resource element. The operation sub-element of object has type attribute to denote whether the operation is create, destroy, initialize, input, change or output. If the operation is initialize, input, change or output, one or more parameter elements are required to record the values.

The VE script is stored in VEKR as XML format, so we can transform the VE process data into human-readable reports on demand. The metadata of the VE process needs to be registered and stored into the VEMR. Once registered, this new process can be used as a service in its own right or as a part of a more complex VE process.

Resource Allocation Service (RAS)

This service is used to find the best mapping between a VE design and available resources, with the goal of satisfying the application requirements (network bandwidth, latency, computing power and storage) and grid constraints. RAS is

directly based on the Globus GRAM services. The selection process is based on the Condor matchmaking algorithm [12], and the resources requests are communicated via a synchronous, query-response protocol based on the ClassAds syntax defined within the Condor project [13]. A ClassAd is a set of expressions that must all evaluate to true in order for a match to succeed.

The location where each service of a VE is executed may have a strong impact on the overall performance of the VE. When dealing with very large data, it is more efficient to keep as much of the computation as near to the data as possible [4]. So the service location is always decided based on the location of preceding services. Another consideration is the compromise between communication and computation [3]. In general, when the computation power is provided by a server, the demand on communications will be high. For example, one virtual experiment uses powerful server to process the requests from the clients and returns the images. This approach generates a lot of traffic on the network and the response time is usually unpredictable. This approach is suitable in situations where the main processing cannot be provided by the client's machine. When sufficient computational power is provided by a local client machine, there will be less demand on communications. To create a reasonably responsive virtual experiment, a compromise has to be made to balance the requirements of communication and computation. A simple allocation algorithm leveraging the above considerations is used to determine the “best” resources as follows.

1. [
2. CompuTime=Typical Execution Time stored in VEMR;
3. CommuTime=inLatency + inData/inBandwidth+ outLatency + outData/inBandwidth;
4. Coex=1.2;
5. ExecTime= CompuTime + CommuTime*coex
6. Rank= 1/ExecTime;
7.]

Line 2 gives the computation time which is estimated as the Typical Execution Time stored in VEMR. Line 3 computes the time needed to transfer the input/output data, where inLatency/outLatency is the network latency of the input/output channel, inData/outData is the amount of the input/output data measured by bit and inBandwidth/outBandwidth is the bandwidth of the input/output channel. Line 4 and 5 gives the value of ExecTime where we give more power to CommuTime because communication time is prone to gain by reason of congestion. Finally rank is the reciprocal of ExecTime, which is the basis for selection.

Data Management Services (DMS)

The DMS is responsible for the search, collection, extraction, transformation and delivery of the data required or produced by the VI, analysis & visualization tools, and constraints computing tools. Data produced by a remote service may be either stored at the same host of the service executed or collected at a central database, or transferred to next service directly. This information is managed by DMS. DMS service is based on the Globus GridFTP, RFT and RLS services. The goal of DMS is

to realize individual warehouse, a single, large, virtual warehouse of a VE data. It deploys a data grid for a VE.

High-Level VE Services Layer
This layer includes services used to search, select and access resources of the VE grid. Moreover, this layer offers services of result visualization. It is the programming interfaces for VE work bench and VEAES developers. Main services are as follows.

VI Access Services (VIAS)
This service is responsible for the search, selection, and deployment of distributed VIs, employing the services provided by VEDS and RAS. The VIs may be simulation software, or remote control physical instruments. The VIs may be implemented as java applet which could be downloaded to the client side, or a web service which will be run at server side or a grid service which will be executed in a Virtual Organization [1]. No matter which kind it is, VI should have standard soft front panel. We don't recommend java applet VIs, because they are difficult to communicate.

Tool Access Services (TAS)
This service is responsible for the search, selection, and deployment of distributed VE tools, employing the services provided by VEDS and RAS. The tools may provide services for data analysis and management, VE constraint computing, and data visualization.

Result Presentation Services (RPS)
Result visualization is a significant step in the VE process that can help students in the VE result interpretation. This service specifies how to generate, present and visualize the data produced by VI and analysis tools. The result could be recorded and stored either as XML format or visualization format.

4 Model of Virtual Experiment Grid

After the general description of the VE Services, here we describe how they are exploited to model the VE grid. Fig.3 shows the different components of the VE grid. In this model teachers and students at the client side could access the resources at the back-end through VE services transparently.

4.1 Clients

The clients are environments for authoring, executing VE and accessing VE Services. A VE Authoring & Executing Tool (VEAET) is offered at client side. VEAET provides services for teachers to design VE plans easily, and for students to execute VE plans. A VE plan is represented by a graph describing resource composition. A node in the plan graph denotes access to one of the distributed resources including VI, tools etc, and a line between nodes describes the interaction and data flows between the services and tools. With this visual tool, a teacher can directly design the VE plan

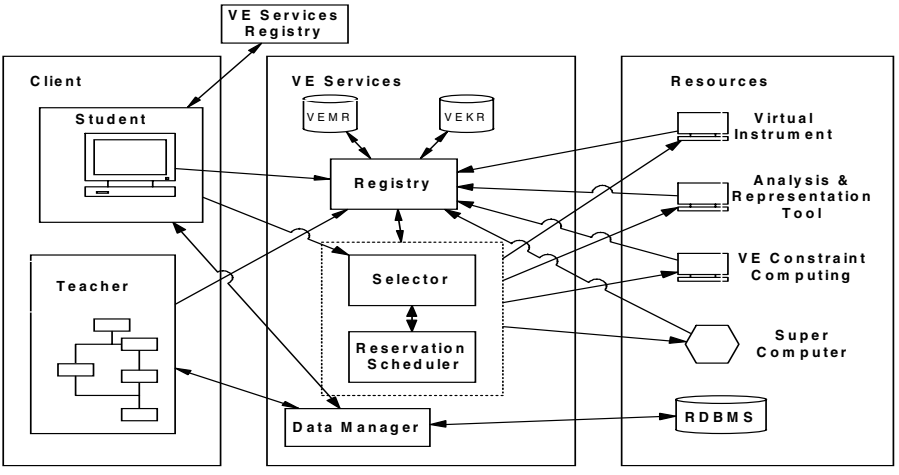

Fig. 3. Model of the Grid Service Based VE

by selecting and dragging. A VE plan could be recorded and stored as XML format locally or published remotely, with the schema figured in Fig.2.

A VE plan could be loaded and executed with VEAET by students anytime and anywhere. Every resource in the VE plan could be mapped and accessed through VI access services, tool access services and result presentation services. When a VE plan is loaded and set to startup, it will firstly get initialized by VEAET. VEAET, acting on the user's behalf, contacts a VE registry that a relevant Virtual Organization maintains to identify VE service providers. The request specifies requirements such as cost, location or performance. The registry returns handles identifying a VE Services that meet user requirements—or perhaps a set of handles representing candidate services. In either case, the user identifies appropriate services. Then VEAET issues requests to the VE services factory specifying details such as the VE operation to be performed, and initial lifetimes for the new VE service instance. Assuming that this negotiation proceeds satisfactorily, a new VE service instance is created with appropriate initial state, resources, and lifetime. The VE service, afterwards, initiates queries against appropriate remote VIs, tools and constraints computing, acting as a client on the user's behalf. Appropriate factories of the relevant resources are selected and then returned from the VE services to the client VEAET. The VEAET is responsible for activating execution on the selected resource as per the scheduler's instruction and then binds the new service instances to the VE plan.

A successful outcome of this process is that a VE plan is transformed into an executable VE. During the execution course, VEAET periodically updates the status of VE execution and records the VE process with the schema in Fig.2 as XML format. Teachers and students could publish a successfully executed VE process through VEDS for further reuse.

4.2 VE Services

The VE services comprise four modules. The registry, with two databases of VE Metadata Repository and VE Knowledge Repository, acts as a VEDS maintaining the registry information of the service providers. The selector uses Grid Resource Information Service to enquire about the dynamic status of resources and uses RAS to select best resources. The reservation scheduler is used to carry out resource reservation. And the data manager is responsible for index, storage and delivery of all the input/output data during the whole VE process.

When the selector receives a request in the form of a ClassAd, it invokes the matchmaking algorithm against the registry representing the available resources, and returns the match list, where the order is determined by the computed ranks, i.e. the "best" match is the first element of the list. The reservation scheduler then makes reservation decisions on behalf of the user. Considering that multiple instances of a resource may be created on a same host, and some resources (such as remote control device) couldn't be accessed by different applications simultaneously, reservation is very important for high-performance VE. The scheduler is based on very simple request-response syntax. A reservation request consists of the computing resources needed such as minimum memory, starting time, the time period and lock information, it returns SUCCESS if the reservation is made, else returns FAILURE and reply with a list of its available resources, available service time etc. If it fails and the user is yet satisfied with the returned parameters, a second time of reservation request will be issued to the same resource with renewed parameters, else a new request will be sent to the next "best" match.

When the "best" match is selected and reserved, the VE plan is ready to run at scheduled time. During the whole run time, the data manager is always at service to index, collect and transfer the input/output data of the VE.

4.3 Resources

The resources of the VE grid include the VI, analysis & visualization tools and VE constraint computing tools, in addition to the ordinary resources such as CPU, memory, and database. These resources may also be implemented upon grid services. For example, considering a VI where a range of sensors produces large volumes of data about the activity of genes in cancerous cells, these data record how each gene responds to the introduction of a possible drug. The analysis to identify potential drugs is both a compute and data intensive task. In order to provide cheap and efficient solutions, grid technology has been used to implement such VI services [11].

5 Conclusion and Future Work

The Grid Services infrastructure is growing up very quickly and is going to be more and more complete and complex both in the number of tools and in the variety of supported applications. In this paper, we propose a virtual experiment model based on novel grid service technology. This model puts forward two-layered virtual experiment services to provide cheap and efficient distributed virtual experiment solution. This model could reuse not only virtual instruments but also compositive

virtual experiments. In order to reuse successfully-deployed virtual experiment, we advance a uniform schema to describe a virtual experiment plan and process. Moreover, we provide a visualized virtual experiment authoring tool for teacher to design an experiment with little effort.

In order for the comprehensive communication between the virtual experiment work bench and other components, future work will focus on further standardization on the virtual instrument interfaces and open VE Services. We will also pay much attention to the improvement of the responsiveness and to the cooperation virtual experiment. Finally we will realize and improve an efficient virtual experiment work bench and hope to see that a rich virtual instrument and tools library gradually come into being.

References

1. Foster et al., The physiology of the grid: An open grid services architecture for distributed systems integration, tech. report, Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.
2. Albert Ip and Ric Canale, A Model for Authoring Virtual Experiments in Web-based Courses, presented at Australasian Society for Computers in Learning in Tertiary Education Conference, 1996.
3. Chuang Liu, Lingyun Yang, Ian Foster and Dave Angulo, Design and Evaluation of a Resource Selection Framework for Grid Applications, Proceedings of the 11th IEEE Symposium on High-Performance Distributed Computing, 2002.
4. Vasa Curcin and Moustafa Ghanem et al., Discovery Net: Towards a Grid of Knowledge Discovery, Knowledge Discovery and Data Mining Conference 2002, ACM 1-58113-567-X/02/0007
5. Foster, What is Grid? A Three Point Checklist, tech. report, http://www.gridtoday.com/02/0722/100136.htm
6. Thomas Sandholm and Jarek Gawor, Globus Toolkit 3 Core- A Grid Service Container Framework, tech. report, Globus project, www-unix.globus.org/ogsa/docs/alpha/gt3_alpha_core.pdf
7. VXI Consortium, http://www.vxibus.org/
8. VPP-2: System Frameworks Specification, VXI plug&play Systems Alliance, http://www-.vxidatacenter.com/news/vxispecs.html,2000.
9. VXI-11.1:TCP/IP-VXIbus Interface Specification, http://www.vxidatacenter.com/news/vxispecs.html,2000
10. VPP-4.1: Virtual Instrument Software Architecture, VXI plug&play Systems Alliance, http://-www.vxidatacenter.com/news/vxispecs.html,2000.
11. Rajkumar Buyya and Kim Branson et al., The Virtual Laboratory: A Toolset for Utilising the World-Wide Grid to Design Drugs, Proceedings of the 2P^{ndP} IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002.
12. M. Livny R. Raman and M. Solomon. Matchmaking: Distributed resource management for high throughput computing. In Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing, Chicago, IL, July 1998.
13. M. J. Litzkow, M. Livny, and M. W. Mutka. Condor—A Hunter of Idle Workstations. In *Proc. of the 8th Int'l Conf.on Distributed Computing Systems*, pages 104–111, 1988.
14. Carnegie Mellon's Virtual Lab, http://www-.ece.cmu.edu/~stancil/virtual-lab/application.html

15. Donald McMullen and Randall Bramley et al., The Xport Collaboratory for High-Brilliance X-ray Crystallography, tech. report, http://www.cs.indiana.edu/ngi/sc2000.
16. Virtual Physics Laboratory, http://jersey.uoregon.edu/vlab/
17. Control The Nuclear Power Plant, http://www.ida.liu.se/~her/npp/demo.html
18. The Interactive Frog Dissection ,http://curry.edschool.virginia.edu/go/frog/
19. Visual Systems Laboratory, http://www.vsl.ist.ucf.edu/
20. Oorange for Experimental Mathematics, http://www-sfb288.math.tu-berlin.de/~konrad/articles/oorange/
21. Mathematical Markup Language (MathML) Version 2.0, http://www.w3.org/TR/MathML2/, 2001