A Web Services Data Analysis Grid*

William A. Watson III†‡, Ian Bird, Jie Chen, Bryan Hess, Andy Kowalski, Ying Chen

Thomas Jefferson National Accelerator Facility

12000 Jefferson Av, Newport News, VA 23606, USA

Summary

The trend in large-scale scientific data analysis is to exploit compute, storage and other resources located at multiple sites, and to make those resources accessible to the scientist as if they were a single, coherent system. Web technologies driven by the huge and rapidly growing electronic commerce industry provide valuable components to speed the deployment of such sophisticated systems. Jefferson Lab, where several hundred terabytes of experimental data are acquired each year, is in the process of developing a web-based distributed system for data analysis and management. The essential aspects of this system are a distributed data grid (site independent access to experiment, simulation and model data) and a distributed batch system, augmented with various supervisory and management capabilities, and integrated using Java and XML-based web services.

KEY WORDS: web services, XML, grid, data grid, meta-center, portal

1. Web Services

Most of the distributed activities in a data analysis enterprise have their counterparts in the e-commerce or business-to-business (b2b) world. One must discover resources, query capabilities, request services, and have some means of authenticating users for the purposes of authorizing and charging for services. Industry today is converging upon XML (eXtensible Markup Language) and related technologies such as SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), and UDDI (Universal Description, Discovery and Integration) to provide the necessary capabilities [1].

The advantages of leveraging (where appropriate) this enormous industry investment are obvious: powerful tools, multiple vendors (healthy competition), and a trained workforce

---

* Work supported by the Department of Energy, contract DE-AC05-84ER40150.

† Correspondence to: William Watson, Jefferson Laboratory MS 16A, 12000 Jefferson Av, Newport News, VA 23606.

‡ Email: Chip.Watson@jlab.org.

(reusable skill sets). One example of this type of reuse is in exploiting web browsers for graphical user interfaces. The browser is familiar, easy to use, and provides simple access to widely distributed resources and capabilities, ranging from simple views to applets, including audio and video streams, and even custom data streams (via plug-ins).

Web services are very much like dynamic web pages in that they accept user-specified data as part of the query, and produce formatted output as the response. The main difference is that the input and output are expressed in XML (which focuses upon the data structure and content) instead of HTML (which focuses upon presentation). The self-describing nature of XML (nested *tag name* + *value* sets) facilitates interoperability across multiple languages, and across multiple releases of software packages. Fields (new tags) can be added with no impact on previous software.

In a distributed data analysis environment, the essential infrastructure capabilities include:

- Publish a data set, specifying global name and attributes
- Locate a data set by global name or by data set attributes
- Submit / monitor / control a batch job against aggregated resources
- Move a data set to / from the compute resource, including to and from the desktop
- Authenticate / authorize use of resources (security, quotas)
- Track resource usage (accounting)
- Monitor and control the aggregate system (system administration, user views)
- (for some applications) Advance reservation of resources

Most of these capabilities can be easily mapped onto calls to web services. These web services may be implemented in any language, with Java servlets being favored by Jefferson Lab (described below).

It is helpful to characterize each of these capabilities based upon the style of the interaction and the bandwidth required, with most operations dividing into low data volume information and control services (request + response), and high volume data transport services (long-lived data flow).

In the traditional web world, these two types of services have as analogs web pages (static or dynamic) retrieved via http, and file transfers via ftp. A similar split can be made to map a data analysis activity onto XML based information and control services (request + response), and a high bandwidth data transport mechanism such as a parallel ftp program, for example bbftp [2]. Other non-file-based high bandwidth I/O requirements could be met by application specific parallel streams, analogous to today's various video and audio stream formats.

The use of web services leads to a traditional three tier architecture, with the application or browser as the first tier. Web services, the second tier, are the integration point, providing access to a wide range of capabilities in a third tier, including databases, compute and file resources, and even entire grids implemented using such toolkits as Condor [3], Legion[4], Globus[5] (see Figure 1).

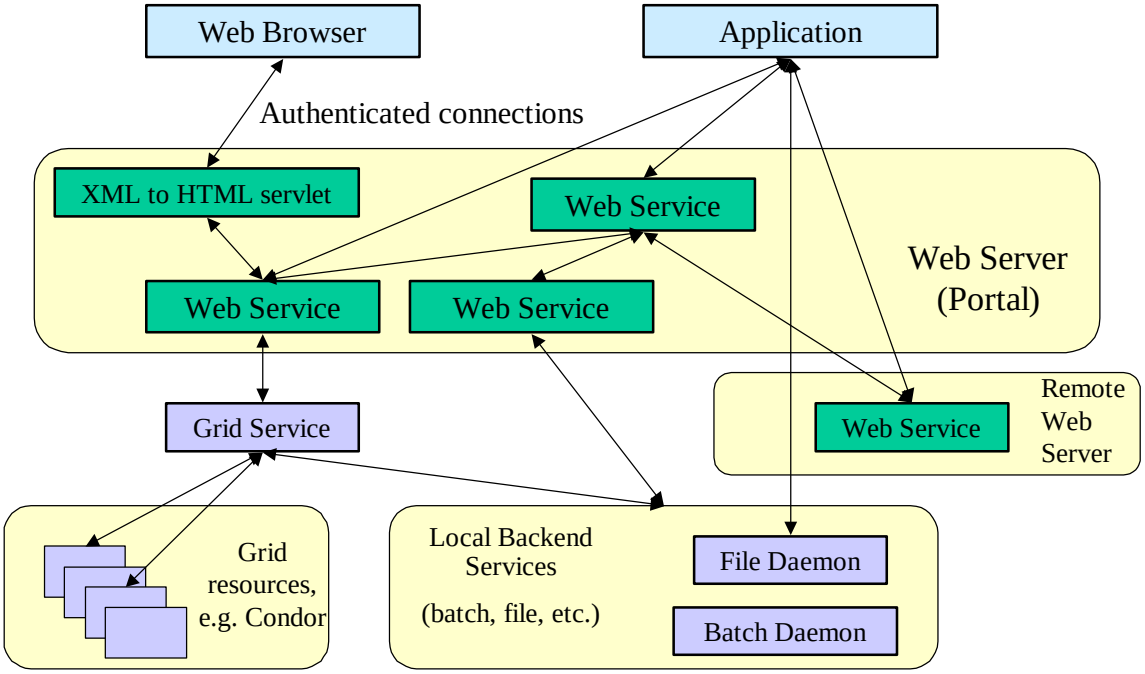

Figure 1: A three tier web services architecture

As an example, in a simple grid portal, one uses a single web server (the portal) to gain access to a grid of resources "behind" the portal. We are proposing a flexible extension of this architecture in which there may be a large number of web servers, each providing access to local resources or even remote services, either by using remote site web services or by using a non-web grid protocol.

All operations requiring privileges use X.509 certificate based authentication and secure sockets, as is already widely used for e-commerce. Certificates are currently issued by a simple certificate authority implemented as a java servlet plus OpenSSL scripts and username + password authentication. These certificates are then installed in the user's web browser, and exported for use by other applications to achieve the goal of "single sign-on". In the future, this prototype certificate authority will be replaced by a more robust solution to be provided by another DOE project. For web browsing, this certificate is used directly. For applications, a temporary certificate (currently 24 hours) is created as needed and used for authentication. Early versions of 3rd party file transfers supports credential forwarding of these temporary certificates.

2. Implementation: Data Analysis Requirements

The Thomas Jefferson National Accelerator Facility (Jefferson Lab) is a premier nuclear physics research laboratory engaged in probing the fundamental interactions of quarks and

gluons inside the nucleus. The 5.7 GeV continuous electron beam accelerator provides a high quality tool for up to three experimental halls simultaneously. Experiments undertaken by a user community of over eight hundred scientists from roughly 150 institutions from around the world acquire as much as a terabyte of data per day, with data written to a 12000 slot StorageTek silo installation capable of holding a year's worth of raw, processed, and simulation data.

First pass data analysis (the most I/O intensive) takes place on a farm of 175 dual processor Linux machines. Java-based tools (JASMine and JOBS, described below) provide a productive user environment for file migration, disk space management, and batch job control at the laboratory. Subsequent stages of analysis take place either at the Lab or at university centers, with off-site analysis steadily increasing. The Lab is currently evolving towards a more distributed, web-based data analysis environment which will wrap the existing tools into web services, and add additional tools aimed at a distributed environment.

Within a few years, the energy of the accelerator will be increased to 12 GeV, and a fourth experimental hall (Hall D) will be added to house experiments which will have ten times the data rate and analysis requirements of the current experiments. At that point, the laboratory will *require* a multi-tiered simulation and analysis model, integrating compute and storage resources situated at a number of large university partners, with raw and processed data flowing out from Jefferson Lab, and simulation and analysis results flowing into the lab.

Theory calculations are also taking a multi-site approach – prototype clusters are currently located at Jefferson Lab and MIT for lattice QCD calculations. MIT has a cluster of 12 quad-processor alpha machines (ES40s), and will add a cluster of Intel machines in FY02. Jefferson Lab plans to have a cluster of 128 dual Xeons (1.7+ GHz) by mid FY02, doubling to 256 duals by the end of the year. Other university partners are planning additional smaller clusters for lattice QCD. As part of a 5 year long range national lattice computing plan, Jefferson Lab plans to upgrade the 0.5 teraflops capacity of this first cluster to 10 teraflops, with similar capacity systems being installed at Fermilab and Brookhaven, and smaller systems planned for a number of universities.

For both experiment data analysis and theory calculations the distributed resources will be presented to the users as a single resource, managing data sets and providing interactive and batch capabilities in a domain specific meta-facility.

3. The Lattice Portal

Web portals for science mimic their commercial counterparts by providing a convenient starting point for accessing a wide range of services. Jefferson Lab and its collaborators at MIT are in the process of developing a web portal for the Lattice Hadron Physics Collaboration. This portal will eventually provide access to Linux clusters, disk caches, and tertiary storage located at Jefferson Lab, MIT, and other universities. The Lattice Portal is being used as a prototype for a similar system to serve the needs of the larger Jefferson Lab

experimental physics community, where FSU is taking a leading role in prototyping activities.

The two main focuses of this portal effort are (1) a distributed batch system, and (2) a data grid. The MIT and JLab lattice clusters run the open source Portable Batch System (PBS) [6]. A web interface to this system [7][8] has been developed which replaces much of the functionality of the tcl/tk based gui included with openPBS. Users can view the state of batch queues and nodes without authentication, and can submit and manipulate jobs using X.509 certificate based authentication.

The batch interface is implemented as Java servlets using the Apache web server and the associated Tomcat servlet engine [9]. One servlet periodically obtains the state of the PBS batch system, and makes that available to clients as an XML data structure. For web browsing, a second servlet applies a style sheet to this XML document to produce a nicely formatted web page, one frame within a multi-frame page of the Lattice Portal. Applications may also attach directly to the XML servlet to obtain the full system description (or any subset) or to submit a new job (supporting, in the future, wide area batch queues or meta-scheduling).

Because XML is used to hold the system description, much of this portal software can be ported to an additional batch system simply by replacing the interface to PBS. Jefferson Lab's JOBS [9] software provides an extensible interface to the LSF batch system. In the future, the portal software will be integrated with an upgraded version of JOBS, allowing support for either of these back end systems (PBS or LSF).

The portal's data management interface is similarly implemented as XML servlets plus servlets that apply style sheets to the XML structures for web browsing (Figure 2.).

The replica catalog service tracks the current locations of all globally accessible data sets. The back end for this service is an SQL database, accessed via JDBC. The replica catalog is organized like a conventional file-system, with recursive directories, data sets, and links. From this service one can obtain directory listings, and the URL's of hosts holding a particular data set. Recursive searching from a starting directory for a particular file is supported now, and more sophisticated searches are envisaged.

A second service in the data grid (the grid node) acts as a front end to one or more disk caches and optionally to tertiary storage. One can request files to be staged into or out of tertiary storage, and can add new files to the cache. Pinning and un-pinning of files is also supported. For high bandwidth data set transfers, the grid node translates a global data set name into the URL of a file server capable of providing (or receiving) the specified file. Access will also be provided to a queued file transfer system that automatically updates the replica catalog.

While the web services can be directly invoked, a client library is being developed to wrap the data grid services into a convenient form (including client-side caching of some results, a significant performance boost). Both applet and stand-alone applications are being developed above this library to provide easy-to-use interfaces for data management, while also testing the API and underlying system.

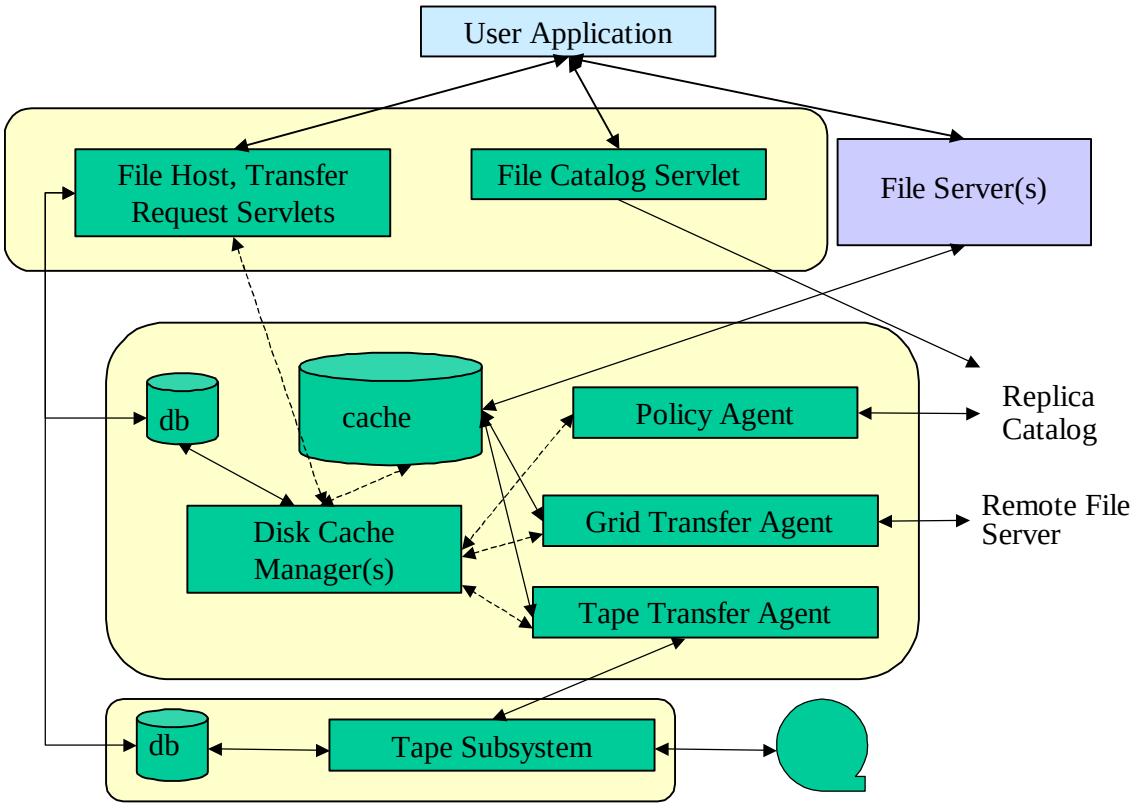

Figure 2: Data Grid Services

The back end services (JASMine [11] disk and silo management) used by the data web services are likewise written in Java. Using Java servlets and web services allowed a re-use of this existing infrastructure and corresponding Java skills. The following is a brief description of this java infrastructure that is being extended from the laboratory into the wide area web by means of the web services described above.

4. Java Infrastructure

4.1. JASMine

JASMine is a distributed and modular mass storage system developed at Jefferson Lab to manage the data generated by the experimental physics program. Originally intended to manage the process of staging data to and from tape, it is now also being applied for user accessible disk pools, populated by user's requests, and managed with automatic deletion policies.

JASMine was designed using object-oriented software engineering and was written in Java. This language choice facilitated the creation of rapid prototypes, the creation of a component based architecture, and the ability to quickly port the software to new platforms.

Java's performance was never a bottleneck since disk subsystems, network connectivity, and tape drive bandwidth have always been the limiting factors with respect to performance. The added benefits of garbage collection, multithreading, and the JDBC layer for database connectivity have made Java an excellent choice.

The central point of management in JASMine is a group of relational databases that store file-related meta-data and system configurations. MySQL is currently being used because of its speed and reliability; however, other SQL databases with JDBC drivers could be used.

JASMine uses a hierarchy of objects to represent and organize the data stored on tape. A single logical instance of JASMine is called a store. Within a store there may be many storage groups. A storage group is a collection of other storage groups or volume sets. A volume set is a collection of one or more tape volumes. A volume represents a physical tape and contains a collection of bitfiles. A bitfile represents an actual file on tape as well as its meta-data. When a file is written to tape, the tape chosen comes from the volume set of the destination directory or the volume set of a parent directory. This allows for the grouping of similar data files onto a common set of tapes. It also provides an easy way to identify tape volumes that can be removed from the tape silo when the data files they contain are no longer required.

JASMine is composed of many components that are replicated to avoid single points of failure: Request Manager handles all client requests, including status queries as well as requests for files. A Library Manager manages the tape. A Data Mover manages the movement of data to and from tape.

Each Data Mover has a Dispatcher that searches the job queue for work, selecting a job based on resource requirements and availability. A Volume Manager tracks tape usage and availability, and assures that the Data Mover will not sit idle waiting for a tape in use by another Data Mover. A Drive Manager keeps track of tape drive usage and availability, and is responsible for verifying and unloading tapes.

The Cache Manager keeps track of the files on the stage disks that are not yet flushed to tape and automatically removes unused files when additional disk space is needed to satisfy requests for files. This same Cache Manager component is also used to manage the user accessible cache disks for the Lattice Portal. For a site with multiple disk caches, the Cache Managers work collaboratively to satisfy requests for cached files, working essentially like a local version of the replica catalog, tracking where each file is stored on disk (Figure 3).

The Cache Manager can organize disks into disk groups or pools. These disk groups allow experiments to be given a set amount of disk space for user disk cache – a simple quota system. Different disk groups can be assigned different management (deletion) policies. The management policy used most often is the least recently used policy. However, the policies are not hard coded, and additional management policies can be added by implementing the policy interface.

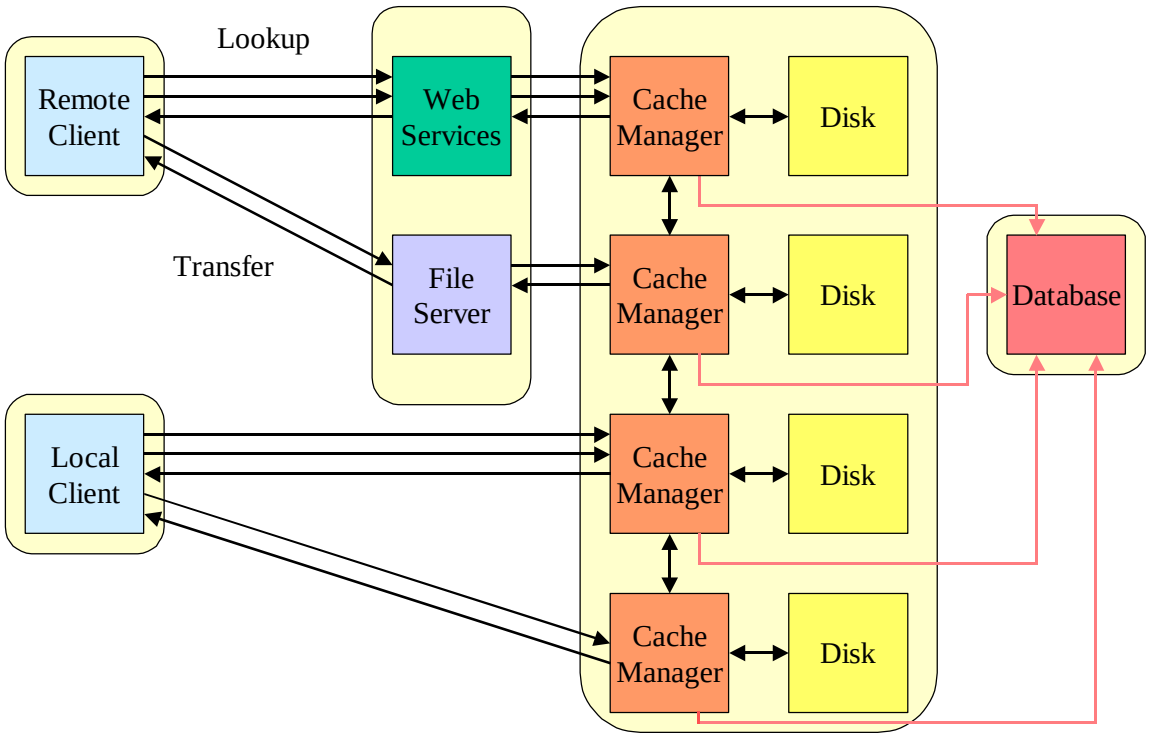

Figure 3: JASMine + Web Services

4.2. JobServer

The Jefferson Lab Offline Batch System (JOBS, or just "the JobServer") is a generic user interface to one or more batch queuing systems. The JobServer provides a job submission API and a set of user commands for starting and monitoring jobs independent of the underlying system. The JobServer currently interfaces with Load Sharing Facility (LSF). Support for other batch queuing systems can be accomplished by creating a class that interfaces with the batch queuing system and implements the batch system interface of the JobServer.

The JobServer has a defined set of keywords that users use to create a job command file. This command file is submitted to the JobServer, where it is parsed into one or more batch jobs. These batch jobs are then converted to the format required by the underlying batch system and submitted. The JobServer also provides a set of utilities to gather information on submitted jobs. These utilities simply interface to the tools or APIs of the batch system and return the results.

Batch jobs that require input data files are started in such a way as to assure that the data is pre-staged to a set of dedicated cache disks before the job itself acquires a run slot and is started. With LSF, this is done by creating multiple jobs with dependencies. If an underlying batch system does not support job dependencies, the JobServer can pre-stage the data before submitting the job.

5. Current Status and Future Developments

The development of the data analysis web services will proceed on two fronts: (1) extending the capabilities that are accessible via the web services, and (2) evolving the web services to use additional web technology.

On the first front, the batch web services interface will be extended to include support for LSF through the JOBS interface described above, allowing the use of the automatic staging of data sets which JOBS provides (current web services support only PBS). For the data grid, policy based file migration will be added above a queued (third party) file transfer capability, using remote web services (web server to web server) to negotiate transfer protocols and target file daemon URL's.

On the second front, prototypes of these web services will be migrated to SOAP (current system uses bare XML). Investigations of WSDL and UDDI will focus on building more dynamic ensembles of web-based systems, moving towards the multi-site data analysis systems planned for the laboratory.

6. Relationship to Other Projects

The web services approach being pursued by Jefferson Lab has some overlap with the grid projects in the Globus and Legion toolkits, Condor, and with the Unicore product [12]. Each seeks to present a set of distributed resources to client applications (and users) as a single integrated resource. The most significant difference between Jefferson Lab's work and these other products is the use of web technologies to make the system open, robust and extensible. Like Unicore, the new software is developed almost entirely in Java, facilitating easy integration with the Lab's existing infrastructure. However, the use of XML and HTTP as the primary application protocol makes the web services approach inherently multi-language and open, whereas Unicore uses a Java-only protocol. At this early stage, the new system does not cover as wide a range of capabilities (such as the graphical complex job creation tool in Unicore or the resource mapping flexibility in Condor-G), but is rapidly covering the functionality needed by the laboratory. In particular, details it contains capabilities considered essential by the laboratory and not yet present in some of the alternatives (for example, the Globus Replica Catalog does not yet have recursive directories, which are now planned for a future release). In cases where needed functionality can be better provided by one of these existing packages, the services of these systems can be easily wrapped into an appropriate web service. This possibility also points towards the use of web service interfaces as a way of tying together different grid systems. In that spirit, Jefferson Lab is collaborating with the Storage Resource Broker [13] team at SDSC to define common web service interfaces to data grids. SRB is likewise developing XML and web based interfaces to their very mature data management product.

ACKNOWLEDGEMENTS

Portions of the Lattice Portal software is being developed as part of Jefferson Lab's work within the Particle Physics Data Grid Collaboratory [14], a part of the DOE's Scientific Discovery Through Advanced Computing initiative.

REFERENCES

[1] For additional information on web services technologies, see (July 9, 2001) http://www.w3.org/TR/2000/REC-xml-20001006 *Extensible Markup Language (XML) 1.0 (Second Edition)* W3C Recommendation 6 October 2000; http://www.w3.org/TR/SOAP/ *Simple Object Access Protocol (SOAP) 1.1* W3C Note 08 May 2000; http://www.w3.org/TR/wsdl *Web Services Description Language (WSDL) 1.1* W3C Note, 15 March 2001; http://www.uddi.org/

[2] See http://doc.in2p3.fr/bbftp/ (July 9, 2001). bbftp was developed by Gilles Farrache (farrache@cc.in2p3.fr) from IN2P3 Computing Center, Villeurbanne (FRANCE) to support the BaBar high energy physics experiment.

[3] Michael Litzkow, Miron Livny, and Matt Mutka, Condor - A Hunter of Idle Workstations *Proceedings of the 8th International Conference of Distributed Computing Systems*, June, 1988; see also http://www.cs.wisc.edu/condor/

[4] Michael J. Lewis, Andrew Grimshaw. The Core Legion Object Model *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing*, August 1996.

[5] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputing Application*s. 11(2):115-128, 1997

[6] See: http://www.openpbs.org/. (July 9, 2001) The Portable Batch System (PBS) is a flexible batch queueing and workload management system originally developed by Veridian Systems for NASA.

[7] See http://lqcd.jlab.org/. (July 9, 2001)

[8] P. Dreher, MIT W. Akers, J. Chen, Y. Chen, C. Watson, Development of Web-based Tools for Use in Hardware Clusters Doing Lattice Physics Proceedings of the Lattice 2001 Conference, to be published (2002) Nuclear Physics B.

[9] See http://jakarta.apache.org/tomcat/ (July 9, 2001)

[10] I Bird, R Chambers, M Davis, A Kowalski, S Philpott, D Rackley, R Whitney, Database Driven Scheduling for Batch Systems, *Computing in High Energy Physics Conference* 1997.

[11] Building the Mass Storage System at Jefferson Lab *Proceedings of the 18th IEEE Symposium on Mass Storage Systems* (2001).

[12] See http://www.unicore.de/ (July 9, 2001)

[13] See http://www.npaci.edu/DICE/SRB/

[14] See http://www.ppdg.net/. (July 9, 2001)