

Indexing and Searching Tera-Scale Grid-Based Digital Libraries

Robert Sanderson
Department of Computer Science
University of Liverpool
Liverpool, L69 3BX, U.K.
Email: azaroth@liverpool.ac.uk

Ray R. Larson
School of Information Management
and Systems
University of California, Berkeley
Berkeley, California, USA, 94720-4600
Email: ray@sims.berkeley.edu

Abstract—The University of California, Berkeley and the University of Liverpool in conjunction with the San Diego Supercomputer Center, are developing a framework for Grid-Based Digital Library systems and Information Retrieval Services (Cheshire3) that operates in both single-processor and distributed computing environments. In this paper we discuss some results of testing Grid-based parallel approaches in indexing and retrieval for a variety of information resources, ranging from small test collections like the TREC and INEX collections, to medium-scale metadata collections like Medline and a test version of University of California Online Union Catalog, MELVYL (with 15 million and 9.5 million records respectively) ranging up to large-scale collections like the US National Records and Archives Administration (NARA) Preservation Prototype. This paper examines our approaches to indexing and retrieving from these collections and the architecture of the system that supports them.

I. INTRODUCTION

The development of “DataGrid” services like the Storage Resource Broker (SRB)[1] open the potential for very large scale information storage and retrieval environments. One area that is rapidly increasing in importance (as well as in the scale of resources required) is that of digital libraries. Digital library projects throughout the world have been creating very extensive repositories of digital information on a wide range of topics. The contents of digital libraries today include collections of scientific data, digitised versions of classical and modern texts in the original languages, commentaries on those texts, historical and current social science survey information, collections of digitised video and audio, and collections of images - including digitised paintings, sculpture, drawings, and other museum objects. In short, the cultural heritage of the world is rapidly being replicated in digital form. Preservation and access to these materials increasingly requires scalable digital libraries, often exceeding the computational resources of single institutions.

Our recent research in designing and developing digital library services has been focused on approaches to indexing and searching in this steadily increasing range of genres and materials. Our primary research focus is information retrieval from large-scale distributed digital libraries in a Grid environment. A close second is research and prototyping in the area of very long term digital preservation, and how grid-scale

information retrieval systems can interoperate with petabytes of diverse data stored over many years.

In order for Information Retrieval (IR) in the evolving Grid parallel distributed computing environment[2] to work effectively, there must be a single flexible and extensible series of “Grid Services” with identifiable objects and a known API to handle the IR functions needed for digital libraries and other retrieval tasks. In this paper we discuss how the Cheshire3 system builds upon the Cheshire IR research project[3] to define and implement a set of objects with precisely defined roles that permit digital library oriented operations to be distributed over many nodes on a network, vastly increasing the throughput of data for compute and storage intensive processes with little overhead beyond single processor solutions.

A recent invitational workshop on “Grid-based Digital Libraries” held at UC Berkeley, defined the potential of using Grid technology to address current issues in the development of digital libraries. The conclusions reached there suggested that Grid technologies can contribute to increasing efficiency of the back-end functions of digital libraries such as data backup, automated replication, and archiving. Data curation systems could float on top of localised storage. Use of Grid technology could facilitate federated digital libraries across institutions. It was concluded that much of the current research on digital library infrastructure could be translated into appropriate architectures for Grid-based systems.

In this paper we describe our recent research in Grid-Based Digital Libraries and Information Retrieval from those libraries. In the next section we discuss the basic architecture and object structure of the Cheshire3 system and how it works in the Grid environment. We will then describe and discuss our experiments on indexing and retrieval using large DL workloads running on both a small cluster and the SDSC TeraGrid system.

II. CHESHIRE3 DESIGN AND ARCHITECTURE

The Cheshire3 system uses an object-oriented design with a hierarchy consisting of two main object types: those objects that represent data and storage, and those that represent processes. The main data objects in the system are shown graphically in Figure 1.

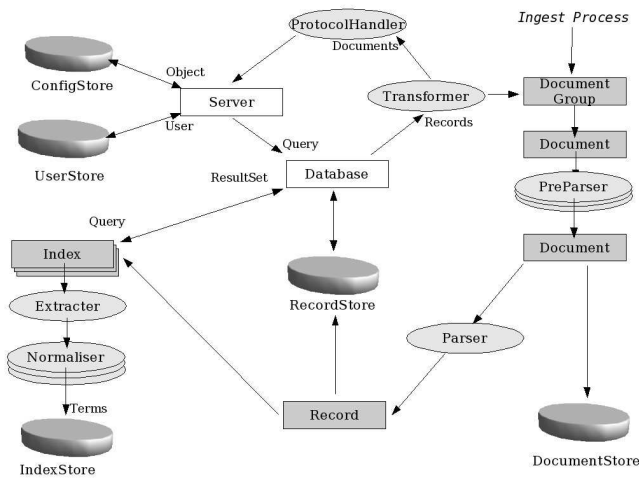


Fig. 1. Cheshire3 Object Model

All objects which are not user-supplied data or the results of a process are configured via XML records. These records include the type of object to instantiate, an identifier for it, as well as the parameters or other information necessary to allow the component to perform its task. With only a minimal set of bootstrapping objects, these configuration records can then be stored, accessed and manipulated in the same way as any other record within the system, including distribution via IR protocols such as OAI-PMH[4] for bulk harvesting or SRW/U[5] for search and retrieval. This allows resources and configurations to be shared seamlessly between distributed instances of the Cheshire3 framework. The objects include:

<i>DocumentGroup</i>	A set of Documents.
<i>Document</i>	Unparsed data representing a single item.
<i>Record</i>	Parsed XML based data representing a single item.
<i>Query</i>	A CQL query parse tree.
<i>ResultSet</i>	An ordered set of symbolic pointers to Records.
<i>Index</i>	An ordered list of terms extracted from one or more Records
<i>User</i>	An authenticated user of the system.

Storage facilities exist for each of these object classes, with a standard API for retrieving the raw data for documents, parsed XML for records (including object configurations), and the instantiated objects themselves.

In addition there are several processing objects including:

<i>PreParser</i>	Convert a Document into another type of Document
<i>Parser</i>	Convert a Document into a Record
<i>Transformer</i>	Convert a Record into a Document
<i>Extractor</i>	Extract data of a given format or type for indexing
<i>Normaliser</i>	Convert data from one format or type to another
<i>ProtocolHandler</i>	Take a request in a known protocol and convert it to an internal representation.

Also, there are three abstract objects:

<i>Server</i>	Logical collection of databases and objects
<i>Database</i>	Logical collection of records and indexes
<i>Workflow</i>	Object that can take input, step through a user-defined sequence of processing steps and produce output.

A. Workflows

For successful Grid-based IR, it is essential to not require the distribution of lines of user written code amongst the nodes doing the processing in order to carry out tasks which have been fine-tuned or configured for a particular document collection. The last abstract object above is the component that allows the Cheshire3 system to work effectively in a distributed environment. Workflow objects are configured like any other object within the system, and hence share their ease of portability. Each workflow contains a series of instructions in its configuration and, when the object is requested, these are compiled dynamically into executable code. Thus the requirements for the system operations are specified by describing the logical flow of data through the various types of processing objects, and no additional programming is needed. Specification of the workflow is done via object identifiers and defaults for the base object types. Workflows can also call other workflows as required, allowing users to set up a hierarchy of modular processes rather than one monolithic specification.

Each object in the system has a (usually unique) identifier as its reference. However, it is possible to re-use identifiers at lower levels of the system, allowing a database to override a particular server object without requiring changes to code or configurations referring to it, for example. The utility of this becomes apparent in the context of distributed workflows, as not only can one workflow call another, one very high level workflow might interact in the same way with many different database-specific workflows, each of which performs customised operations to achieve the same end result.

The workflow specification does not make use of a standard workflow language, such as WS-BPEL[6], but instead the implementation has its own very simple set of instructions specific to the Cheshire3 architecture and use cases. Work is underway to integrate the Kepler[7] workflow engine with Cheshire3, both with a very high level interface (using the SRW web service specification for information retrieval) and a per object level interface, duplicating the existing workflow functionality. We also intend to have a Cheshire3 to Kepler interface which will allow Kepler workflows to be run as any Cheshire3 processing object.

The instructions in the Cheshire3 workflow are represented as nested XML elements in the workflow object's configuration entry. These instructions are currently:

<i>for-each</i>	Iterate through a group of objects
<i>fork</i>	Split the input data between multiple paths
<i>assign</i>	Assign a value from one location to another
<i>object</i>	Call an object method with the current data
<i>try/except/raise</i>	Error condition handling
<i>break/continue</i>	Loop control handling
<i>return</i>	Process flow handling

This allows workflows to be rapidly built up in XML

without requiring special tools. The workflow objects are capable of processing the entire ingestion phase from document acquisition through preParsing, parsing, term extraction, normalisation and storing. The advantage to the current workflow system, compared to the one described in previous papers[8], is that it allows indexing processes to be defined either per index (the workflow stepping through each index object) or per access point in the document (the workflow stepping through each XPath source). For example, one might desire many different indexes on a document's title: exact title, normalised to remove the leading article, case normalised, keyworded, stemmed keywords, tagged for part of speech and then noun phrases and verbs extracted, those phrases and words stemmed, and so forth. When the indexing process includes part of speech tagging or other computationally expensive process, then the processes must be performed in the most efficient way possible, and the expensive tasks certainly not repeated.

Once the distributed infrastructure has been built, one or more "master" workflows divide the processing requirements among "slave" processes. When a slave process has completed its assigned workflow, it may return results to the master to be merged. As object interactions within the architecture are well defined, the result from any workflow is also well defined, based on the last processing object. This is very important for trees or graphs of distributed workflows.

In the current implementation the communication is abstracted from the system by a TaskManager object. This object is responsible for sending requests across the grid or cluster and receiving back the response from the remote node. To date, two implementations have been written; one for the Parallel Virtual Machine[9] (PVM) protocol, and the other for the Message Passing Interface[10] (MPI). Although the performance is expected to suffer, a SOAP[11] based implementation is currently being developed to enable object level interaction via a web service based interface. Each node instantiates a TaskManager for the appropriate protocol, not just the master workflow. This allows slave nodes to interact with each other, rather than just with the master. For example, a node might be dedicated to caching and storing parsed records which it is given by other slave nodes in the system.

While the master nodes are waiting for the slave nodes to process the data, they are not left idle but instead are fetching the next set of documents to be processed. Instead of each request being a single document to process, it is common to have a variable number of documents (represented as a documentGroup) sent in a request to ensure that size of the request message is appropriate for the infrastructure. Too many small requests (and hence responses) can result in the distributed system spending more time serialising and deserialising than fewer, larger messages. In order for this to occur, the requests must be asynchronous – the master cannot block until it receives a response.

Each database is a logical rather than physical collection of records and hence it can be easily divided amongst many nodes or combined at a single location. This means that each node on the cluster can either look after a slice of the database

or do the processing required and then return the record for central storage. The same applies to indexing to a lesser extent; if the index files at each node store all of the terms for the records at the node, then interacting with the sub database is very fast, but the intermediate resultsets will need to be merged before a global query can be answered. If each node maintains a portion of the terms for all of the records, then it can answer the query authoritatively, however this requires some central authority to manage the division of the index terms.

B. Relation to Other Architectures

This architecture is specified at a much lower level than that of, for example, the Grid-IR working group[12]. Nassar, Dovey et al. propose in their 2003 paper only three main classes of service, a Collection Manager to harvest or import items, an Indexing and Searching module, and a Query Processor to take care of distributed searches. The Index/Search module is the most interesting, as it is based on the Z39.50 model. As Cheshire3 exposes the SRW web service based profile of Z39.50, it could be considered an implementation of the Grid-IR IS module. Cheshire3 does not in any way compete with the Grid-IR specification, it simply goes beyond it to expose a lower level of functionality.

The DILIGENT project in the EU aims to build a test-bed that integrates Grid and Digital Library technologies[13]. Their architecture is again at a higher level than that of the Cheshire3 architecture, including such aspects as "Dynamic Virtual Organisation Support" and Process Design modules. As with the Grid-IR system, Cheshire3 could likely be used as an implementation within the "Index and Search" section of the DILIGENT architecture. Alternatively, many of the objects in the DILIGENT architecture could be redefined as mixtures of Cheshire3 objects. For example an annotation service is simply another database where the records are related to documents, and this can obviously be represented in the Cheshire3 object model. By having a very low level specification, we are able to easily configure instances of the architecture to duplicate or interoperate with other systems.

III. DATAGRID INTEGRATION

In order to maintain many gigabytes or terabytes of data, a scalable, distributed storage system is essential. To also enable the system to act as a digital preservation environment, it should allow multiple redundant copies of each document, further extending the storage requirements. The San Diego Supercomputer Center has developed a data grid scale system called the Storage Resource Broker (SRB) which fits this description and is in widespread use around the world.

The SRB abstracts the interaction with the physical storage system in order to seamlessly maintain its information in a variety of locations, including regular file systems, relational databases, Storage Area Networks or even via transfer protocols such as FTP. The system allows for transparent, policy controlled replication of the data as it is ingested to multiple physically distinct sites.

The SRB also maintains the metadata associated with the file in a metadata catalog system (MCAT) to allow the different storage systems underneath to be abstracted into a common API, and for the data to be retrieved from an appropriate location. The MCAT can be implemented in any relational database management system, however Oracle and PostgreSQL have been recommended and tested.

As with the grid based processing power of the Cheshire3 architecture, although the SRB is designed to run across distributed sites to ensure the preservation and availability of the data, it is very possible to run it with only one available machine.

The SRB has been integrated into the Cheshire3 system in several different data acquisition and storage roles. A `documentGroup` interface allows Cheshire3 to traverse SRB collections and retrieve data to be processed. This allows us to maintain huge datasets, such as the NSDL persistent web crawl, as well as multiple large datasets, such as the MELVYL and Medline collections, in a single abstract location but at the same time permitting fast access from any remote site. The `documentGroup` interface then acts as an intermediary between the SRB and Cheshire3 systems.

More importantly, `documentStore` and `recordStore` implementations allow us to store acquired and processed data remotely for future access, with the same advantages as for the input data sets. For configurations which include data mining or text mining processes such as part of speech tagging, it is important to not only store the original document and the final resulting parsed XML representation, but also the documents created at each step of the process. As each processing object has an atomic operation, all of the intermediary results can be stored in the `datagrid` for future retrieval and reprocessing. For example, if a new word stemming process is implemented for medical data such as the Medline collection, it would be very wasteful to have to redo all of the part of speech tagging. The storage capabilities of the SRB enable us to store the results of every transformation that occurs within the system.

The final storage requirement to be considered for the Cheshire3 system is maintaining the inverted indexes and document vector files. Currently these files must be interacted with via a local interface as the implementation relies on the Berkeley DB storage system[14]. These database files, once constructed, could be stored within the SRB and retrieved when the `indexStore` is instantiated, however this doesn't obviate the need for local storage space sufficient to maintain them. Dividing the index files into smaller chunks for storage in the data grid and only retrieving the sections required is a possibility currently being explored.

The SRB is being integrated into other systems as well, such as the DSpace Digital Library Framework[15]. In this scenario, the data can be administered and curated via the DSpace interface, but Cheshire3 can access it directly via the underlying SRB implementation. In this way we will soon, and with only minimal implementation work, have an advanced indexing and search system for DSpace/SRB installations.

A. Early Performance Tests

Early performance tests used 1 'master' and 15 slave indexing processes (in a cluster of 8 dual 1.6Ghz Athlon machines with 2Gb of RAM, with gigabit ethernet). Using this configuration on a workflow that parses simple MARC bibliographic records, we are able to achieve a sustained indexing rate of about 10,000 records per second (even when competing with other tasks on the cluster). For larger, more complex, records we tested with 640 Mb of TEI encoded documents, and were able to parse, store and index the entire collection in 3 minutes and 20 seconds.

B. Medline

The TeraGrid[16] cluster at SDSC consists of 256 machines, each with dual 2.4 gigahertz 64 bit Itanium processors and 4 gigabytes of main memory. In order to develop Cheshire3 for the NARA and NSDL datasets (please see the section on Future Work for more information), we have a 'small' development grant of 30 thousand CPU hours. Part of that development work has been to work with different large data sets, two of which will be described below.

For the purposes of the National Centre for Text Mining[17] (NaCTeM) in the UK, we have a copy of the full Medline[18] database of some 15 million article abstracts. NaCTeM is jointly run by the Universities of Liverpool, Manchester and Salford, and the Cheshire3 system provides the information processing and retrieval architecture. In this context it is important to analyse the full text in terms of part of speech tagging, which is computationally expensive. We use a part of speech tagger provided by the University of Tokyo, another member of NaCTeM along with UC Berkeley and the San Diego Supercomputer Centre, integrated within Cheshire3, to fill this role.

When part of speech tagging is performed during indexing, it takes significantly more time than any other process. Professor Jun'ichi Tsujii (University of Tokyo) reports that their cluster of 200 machines took 2 weeks to perform the part of speech tagging alone across the entire data set.

The Medline documents are stored in the SRB as received in chunks of 30,000 records per file. In normal conditions, it takes around 30 seconds to retrieve a file from the SRB (about 1 Mb / second, sustained) and sometimes up to 5 seconds to locate all of the documents within the file. This limits the number of records processed per minute to around 50,000 per process committed to fetching and locating documents from the SRB.

To date, current experiments have only used one task dedicated to writing the parsed records to a `RecordStore`. When run on the Teragrid with significantly more slave processes than is possible on the clusters at Liverpool, this creates an obvious bottleneck. This issue is multiplied by the message reception code in the underlying MPI interface[19] which does not permit a round-robin approach to asynchronous receiving to ensure that all tasks have an equal chance to send a message.

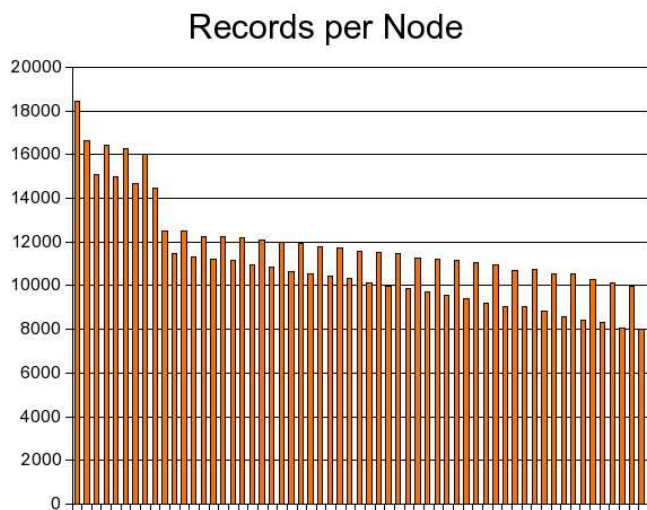


Fig. 2. Records Per Node

By the end of processing 750 thousand records, using one master, one write task and 59 processing slaves, the first slave had processed 21 thousand records, while the last slave had only been able to process 8 thousand. In order to verify that this was not a side effect of sending 300 documents per message, the same processing was done with one record per message, generating very similar results of 19 thousand for the first task and 8 thousand for the last. However more alarming is that the second task was only able to process 17 thousand records. This means that for the same time period, many tasks were sitting idle waiting for the record writing task to accept their messages. The graph in Figure 2 shows the records processed per task after 670 thousand records had been indexed.

The reason for a single storage task is to enable the system to easily assign a unique, consecutive, integer based identifier to each record. This is not required by the architecture (in fact the architecture can use any string serialisable value as an identifier) and experiments to be reported on later will use multiple write tasks.

Once the records had been written to a RecordStore by the above experiment, a further test was performed in which the records were retrieved and indexed. The messages sent across the MPI bridge were groups of 500 identifiers of the records to be processed. Using the same 59 tasks, the bottleneck described above did not appear and all 750 thousand items were able to be indexed in 2 minutes and 52 seconds, thus more than a quarter million records per minute (4360 per second). By the end of the run, all of the tasks had been sent between 9500 and 10500 records to process – a much more balanced distribution. At this rate of processing, the entire collection would be indexed in under an hour, using only around a tenth of the processing resources of the Teragrid cluster at SDSC. Using all the nodes available would mean that indexing could be performed in around 7 minutes. This means that real time query processing on 50+ gigabytes of

data is within grasping range, without any inverted indexes!

To verify that this would scale, a similar run was performed using groups of a thousand record identifiers, distributed to 120 processes. This was completed in one minute, 33 seconds, the extra few seconds likely being attributable to the increased initialisation costs of the extra tasks, but a larger experiment was necessary.

With 1.8 million records, 40 machines, 80 processes and 1000 record identifiers per group, it took four minutes, 45 seconds (6315 records per second). With the same data, 100 machines and 200 processes, it took one minute, 57 seconds (15385 records per second). With two and a half times more processes, given the first run, one would expect somewhere in the vicinity of 15780 records per second if there were no increased overheads for maintaining the tasks. That the difference was under 400 is very encouraging, as it shows that the overhead for tasks is low: 2.5 times as many tasks attained 2.44 times the performance.

C. IR Test Collections

In order to gather data on situations where the processing requires significantly more work per record than plain keyword and exact value indexing, 267 thousand articles from the TREC collection (taken from the Associated Press and Financial Times sub-collections) were normalised to XML, parsed and then processed using the Tokyo part of speech tagger as part of the indexing workflow. This allowed us to investigate the use of natural language processing as a means of determining key phrases and words to index, rather than a simple stoplist approach.

The text was extracted as keywords and stemmed. This generated 1.1 gigabytes of interim index file to be batch loaded. The text was also tagged and keywords extracted without any stemming. Due to the lack of stemming and the extra space required to maintain the part of speech tag for every word, this takes up 1.5 gigabytes for the same set of words. This inflation is expected and allows for queries such as 'fish' as a noun rather than a verb. A stemmed index with only noun and verb forms was then extracted, requiring 667 megabytes. Finally phrase indexes consisting of two or more adjacent adjectives and/or nouns, both stemmed (300 megabytes) and unstemmed (324 megabytes) were generated.

It took 32 machines 35 minutes to do the processing, including writing to the RecordStore. Two runs were performed with almost identical timing results, one using 60 processes and one using 90. The reason for the lack of increased performance is that the part of speech taggers running in each process were competing for the same CPU time. As the majority of the time was spent processing the records with the part of speech tagger, the bottleneck of only one write task was not encountered.

V. FUTURE WORK

Cheshire3 is part of the US National Archives and Records Administration (NARA) Electronic Records Archives testbed[20]. It is being evaluated as part of the information

retrieval aspects of the prototype, along with the SRB and another technology being developed at the University of Liverpool, the Multivalent document platform[21]. The NARA prototype will involve much larger datasets than those reported on above, hence we need to progress from gigabytes into terabytes.

In order to perform experiments of this size, with a view to developing a production system capable of supporting the requirements of NARA, we are in the process of setting up a system to index the National Science Digital Library persistent web crawl archive[22]. This is around 3.5 terabytes of data, across 27 million items. We expect to have positive results to report at the conference on our experiences with this collection.

Further development is constantly taking place on integrating other products within the Cheshire3 environment, including the Kepler workflow system but also data mining, text mining and document processing applications.

In terms of Grid development, we are working on a SOAP based interface to provide a web service-oriented architecture. We expect to be able to report on the results of storing inverted index files in the data grid at the conference, and also to have some initial results for large scale distributed query processing.

VI. CONCLUSIONS

We have presented the Cheshire3 architecture and discussed its use as a distributed large scale information retrieval system. In particular, we have demonstrated its efficiency and speed when running on different architectures with diverse datasets. We have discussed some of the issues that have arisen during the work, such as the processing and document retrieving bottlenecks of working in a very distributed environment, and how they can be resolved or worked around. Finally we have shown the integration of third party data processing tools, and discussed the way forwards for the system. It remains only to say that the Cheshire3 system is available as open source software at <http://www.cheshire3.org/> and relies on only open source tools.

ACKNOWLEDGMENTS

Design of the Cheshire3 system was supported by the NSF and JISC (U.K) under the *International Digital Libraries Program* award #IIS-9975164.

REFERENCES

- [1] A. Rajasekar, M. Wan, R. Moore, W. Schroeder, G. Kremenek, A. Jagatheesan, C. Cowart, B. Zhu, S.-Y. Chen, and R. Olschanowsky, "Storage resource broker - managing distributed data in a grid," *Computer Society of India Journal*, vol. 33, no. 4, pp. 42-54, 2003.
- [2] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, 2nd ed. Amsterdam: Elsevier, 2004.
- [3] R. R. Larson, J. McDonough, P. O'Leary, L. Kuntz, and R. Moon, "Cheshire II: Designing a next-generation online catalog," *Journal of the American Society for Information Science*, vol. 47, no. 7, pp. 555-567, July 1996.
- [4] C. Lagoze, H. Van de Sompel, M. Nelson, S. Warner, "The Open Archives Initiative Protocol for Metadata Harvesting, Version 2.0", June 2002. <http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm>
- [5] R. Denenberg, R. Sanderson, M. Dovey et al. (eds), "SRW - Search/Retrieve Webservice", February 2004. <http://www.loc.gov/srw>

- [6] OASIS WSBPEL Technical Committee, "Web Services Business Process Execution Language". 2005. <http://www.oasis-open.org/committees/wsbpel/charter.php>
- [7] B. Ludscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, Y. Zhao, "Scientific Workflow Management and the Kepler System", in *Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows*, to appear 2005.
- [8] R. R. Larson, R. Sanderson, "Grid-based Digital Libraries: Cheshire3 and Distributed Retrieval", in *Proceedings of the 5th ACM/IEEE Joint Conference on Digital Libraries Digital Libraries: Cyberinfrastructure for Research and Education 2005*, pp.112-113
- [9] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam, "PVM: Parallel Virtual Machine" MIT Press, 1995
- [10] Message Passing Interface Forum. "MPI: A Message-Passing Interface standard (version 1.1)", Technical report, 1995. <http://www.mpi-forum.org>.
- [11] W3C, "SOAP Specifications" June 2003. <http://www.w3.org/TR/soap/>
- [12] N. Nassar, G. Newby, K. Gamiel, M. Dovey, J. Morris, "Grid Information Retrieval Architecture", 2003, [urlhttp://www.gir-wg.org/](http://www.gir-wg.org/)
- [13] DILIGENT Project, "Architectural Overview", April 2005, <http://diligentproject.org/content/view/71/99>
- [14] SleepyCat Software, "Berkeley DB", 2005, <http://sleepycat.com/products/db.shtml>
- [15] L. Declerck, C. Frymann, "DSpace/SRB Integration", *CNI Fall Task Force Meeting*, <http://libnet.ucsd.edu/nara/>
- [16] "About the TeraGrid", 2005. <http://www.teragrid.org/about/index.html>
- [17] "National Centre for Text Mining", 2005. <http://www.nactem.ac.uk>
- [18] National Library of Medicine, "Medline Factsheet". <http://www.nlm.nih.gov/pubs/factsheets/medline.html>
- [19] "PyMPI", 2005, <http://pympi.sourceforge.net/>
- [20] The National Archives, "National Archives Electronic Records Archives Option Award Announcement" 2005. <http://www.archives.gov/era/acquisition/option-award.html>
- [21] T. Phelps, P. Watry, "A No-Compromises Architecture for Digital Document Preservation" in *Research and Advanced Technology for Digital Libraries 9th European Conference, ECDL2005, Proceedings 2005*, pp. 266-277
- [22] P. Tooby, "Building a 'Memory' for the National Science Digital Library", in *NPACI Online* vol. 7, issue 2, January 2003. <http://www.npaci.edu/online/v7.2/nsdl.html>