DELOS
Network of Excellence on Digital Libraries

Future Digital Library Management Systems: System Architecture and Information Access

8th DELOS thematic workshop
Schloss Dagstuhl, Germany
March 29 - April 1, 2005

Editors:

Yannis Ioannidis – University of Athens, Greece (Program Co-Chair)
Hans-Jörg Schek – ETH Zurich, Switzerland and UMIT Innsbruck, Austria (Program Co-Chair)
Gerhard Weikum – MPI Saarbrucken, Germany (General Chair)

DELOS: a Network of Excellence on Digital Libraries

www.delos.info

Information Society Technologies

Table of Contents

Future Digital Library Management Systems:
System Architecture and Information Access

8th International Workshop of the DELOS Network of Excellence on Digital Libraries

Schloss Dagstuhl, Germany
March 29 – April 1, 2005

Compiled by
Yannis Ioannidis (Program Co-Chair)
Hans-Jörg Schek (Program Co-Chair)
Gerhard Weikum (General Chair)

Call for Participation

8th International Workshop of the DELOS Network of Excellence on Digital Libraries

Schloss Dagstuhl, Germany, March 29 - April 1st, 2005

on Future Digital Library Management Systems

System Architecture & Information Access

DELOS is an EU-funded interdisciplinary Network of Excellence on Digital Libraries (www.delos.info) with a broad vision: Future digital libraries (DLs) should enable any citizen to access human knowledge any time and anywhere, in a friendly, multi-modal, efficient and effective way. Achieving this vision requires development of a next generation of Digital Library Management Systems (DLMSs) that will dramatically change Digital Libraries from the form we know them currently. At the forefront of such development are two main issues:

System Architecture
Information Access

This workshop, the 8th in the DELOS series of Thematic Workshops, is devoted to these two critical themes. The objective of the workshop is to bring together researchers and practitioners interested in these two areas and their inter-connections, to identify fundamental system services that allow the development and operation of future Digital Libraries, and to explore the main relevant technical directions.

With respect to system architecture, Peer-to-Peer (P2P) Data Management, Grid Middleware (Grid), and Service-oriented Architecture (SoA) are the topics of primary interest. P2P architectures allow for loosely-coupled integration of information services and sharing of information. Different aspects of P2P systems (e.g., indexes and application platforms) must be combined to achieve the desired functionality. On the other hand, Grid computing middleware is needed because certain services within Digital Libraries are complex and computationally intensive (e.g., for extraction of features in multimedia documents to support content-based similarity search or for information mining in bio-medical data). Finally, SoA provides mechanisms to describe the semantics and usage of information services. In addition, in an SoA, we have mechanisms to combine services into workflow processes for sophisticated search and maintenance of dependencies.

With respect to information access, a fundamental challenge arises from the great variety of content that future Digital Libraries are called to manage, each one with its own characteristics and particularities, with respect to both format and meaning. Organization of information within an individual source and efficient and effective search are key issues that need to be addressed. An additional challenge arises from the user interfaces planned for future systems, where the general trend is towards richer languages and diverse interaction styles both at the syntactic and at the semantic level. New interaction management, optimization, execution, and result consolidation algorithms need to be devised to support the emerging functionalities.

Clearly, the two themes are closely interrelated and some of the most exciting problems arise at the intersection of the two. The goal of this workshop is to provide a forum for discussing the latest advances and on-going efforts in these and related areas as the field moves towards future DLMSs.

program

start: Thuesday, March 29th, 12:00 Lunch
end: Friday, April 1st, 12:00 Lunch

keynote

Peter Buneman
The two Cultures of Digital Curation

invited talks

Stefan Gradmann
Specific Aspects of e-science scenarios in the hermeneutical disciplines

Keith Jeffery
Digital Libraries in a Grid Environment

Michalis Vazirgiannis
Semantic overlay generation in P2P architectures

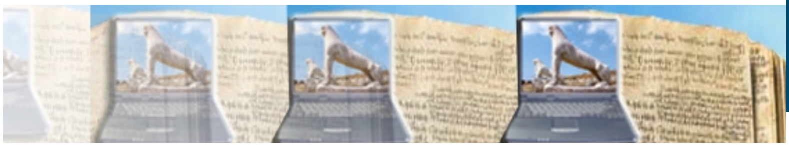
session 1: "Architectural Issues"

Wurz, Schuldt
Dynamically Making Use of Distributed Data Sources in a Grid Environment

Warns, Hasselbring, Roantree
Influence of Replication on Availability within P2P Systems

Bischofs, Giesecke, Hasselbring, Niemann, Steffens
Adaptive replication strategies and software architectures for P2P systems

Mazurek, Werla
Distributed Services Architecture in dLibra Digital Library Framework



8

session 2: "Semantics and Context"
Koffina, Serfiotis, Christophides, Tannen, Deutsch
Integrating XML Data Sources using RDF/S Schemas: The ICS-FORTH Semantic Web Integration Middleware (SWIM)
Kokkinidis, Sidirourgos, Dalamagas, Christophides
Semantic Query Routing and Processing in P2P Digital Libraries
Tryfonopoulos, Idreos, Koubarakis
Publish/Subscribe Functionalities for Future Digital Libraries using Structured Overlay Networks
Meghini, Spyratos
Information Access in Digital Libraries: Steps Towards a Conceptual Schema

session 3: "Search and Indexing I"
Bender, Michel, Weikum, Zimmer
Challenges of Distributed Search Across Digital Libraries
Di Nunzio, Ferro
DIRECT: a Distributed Tool for Information Retrieval Evaluation Campaigns
Barton, Zezula
p-index - An Index for Graph Structured Data
Springmann, Balko, Schek
Efficient and Effective Matching of Compound Patient Records

session 4: "Search and Indexing II"
Damjanovic, Plant, Balko, Schek
User-Adaptable Browsing and Relevance Feedback in Image Databases
Novak, Zezula
Indexing the Distance Using Chord: A Distributed Similarity Search Structure
Meghini, Spyratos
Query Tuning through Refinement / Enlargement in a Formal Context
Nejdl, Paiu
I know I stored it somewhere-Contextual Information and Ranking on our Desktop

Complementing the technical program, there will be a "gong show" like event in an evening session where participants may introduce and discuss own ideas. In addition, there will be administrative sessions on DELOS WP1 and WP2 management issues followed by dedicated working group meetings that are to be held in parallel. Workshop participants are invited to propose ideas in a brainstorming session. The workshop will be concluded by presenting the working group results. A hiking tour to the area surrounding Dagstuhl will be taking place as social event.

Program Committee
- Maristella Agosti (Univ. of Padua, Italy)
- Elisa Bertino (Univ. of Milano, Italy & Purdue Univ, USA)
- Donatella Castelli (CNR-ISTI, Italy)
- Hsinchun Chen (Univ. of Arizona, Tuscon, AZ, USA)
- Stavros Christodoulakis (Technical Univ. of Crete, Greece)
- Bruce Croft (Univ. of Massachussetts, Amherst, MA, USA)
- Alex Delis (Univ. of Athens, Greece)
- Ed Fox (Virginia Tech., VA, USA)
- Michael Fresston (UC Santa Barbara, CA, USA)
- Norbert Fuhr (Univ. of Duisburg, Germany)
- Wilhelm Hasselbring (OFFIS Oldenburg, Germany)
- Yannis Ioannidis (Univ. of Athens, Greece)
- Carlo Meghini (CNR-ISTI, Pisa, Italy)
- Reagan Moore (SDSC, San Diego, CA, USA)
- Erich Neuhold (FhG Darmstadt, Germany)
- Hans-Jorg Schek (ETH Zurich, Switzerland & UMIT Innsbruck, Austria)
- Timos Sellis (Nat. Technical Univ. of Athens, Greece)
- Can Turker (ETH Zurich, Switzerland)
- Heiko Schuldt (UMIT Innsbruck, Austria)
- Stratis Viglas (Univ. of Edinburgh, Edinburgh, Scottland)
- Gerhard Weikum (MPI Saarbrucken, Germany)
- Pavel Zezula (Masaryk Univ. Brno, Czech Republic)

Workshop Organization
Gerhard Weikum, General Chair
Max-Planck Institute of Computer Science
Saarbruecken, Germany
Email: weikum@mpi-sb.mpg.de

Yannis Ioannidis, PC co-Chair
University of Athens
Athens, Greece
Email: yannis@di.uoa.gr

Hans-Jorg Schek, PC co-Chair
ETH Zurich, Switzerland and
UMIT Innsbruck, Austria
Email: hans.joerg.schek@umit.at

Registration
Email: Isabella.Fritz@umit.at

Information
http://www.dagstuhl.de/

Preliminary Schedule

Tuesday, March 29, 2005

12:15 Lunch

13:50 Welcome

14:00 Invited Talk by *Stefan Gradmann*: Specific Aspects of E-Science Scenarios in the Hermeneutical Disciplines

14:30 – 16:00 Session 1: Architectural Issues

Manfred Wurz, Heiko Schuldt
Dynamically Making Use of Distributed Data Sources in a Grid Environment

Timo Warns, Wilhelm Hasselbring, Mark Roantree
Influence of Replication on Availability within P2P Systems

Ludger Bischofs, Simon Giesecke, Wilhelm Hasselbring, Heiko Niemann, Ulrike Steffens
Adaptive replication strategies and software architectures for P2P systems

Cezary Mazurek, Marcin Werla
Distributed Services Architecture in dLibra Digital Library Framework

16:00 Coffee Break

16:30 – 18:00 Session 2: Semantics and Context

Ioanna Koffina, Giorgos Serfiotis, Vassilis Christophides, Val Tannen, Alin Deutsch
Integrating XML Data Sources using RDF/S Schemas: The ICS-FORTH Semantic Web Integration Middleware (SWIM)

George Kokkinidis, Lefteris Sidirourgos, Theodore Dalamagas, Vassilis Christophides
Semantic Query Routing and Processing in P2P Digital Libraries

Christos Tryfonopoulos, Stratos Idreos, Manolis Koubarakis
Publish/Subscribe Functionalities for Future Digital Libraries using Structured Overlay Networks

Carlo Meghini, Nicolas Spyratos
Information Access in Digital Libraries: Steps Towards a Conceptual Schema

18:00 Dinner

20:00 DELOS WP1/WP2 Meeting

Wednesday, March 30, 2005

9:00 **Invited Talk by *Michalis Vazirgiannis*: Semantic Overlay Generation in P2P Architectures**

9:30 – 11:00 **Session 3: Search and Indexing**

Matthias Bender, Sebastian Michel, Gerhard Weikum, Christian Zimmer
Challenges of Distributed Search Across Digital Libraries

Georgio Maria Di Nunzio, Nicola Ferro
DIRECT: a Distributed Tool for Information Retrieval Evaluation Campaigns

Stanislav Barton, Pavel Zezula
p-index - An Index for Graph Structured Data

Michael Springmann, Sören Balko, Hans-Jörg Schek
Efficient and Effective Matching of Compound Patient Records

11:00 **Coffee Break**

11:30 **Discussion of Working Group Topics**

12:15 **Lunch**

14:00 **Joint Hiking Tour**

18:00 **Dinner**

20:00 **DELOS WP1/WP2 Meeting (Continued)**

Thursday, March 31, 2005

9:00-10:00 **Keynote by *Peter Buneman*: The Two Cultures of Digital Curation**

10:00 **Coffee Break**

10:30 – 12:00 **Session 4: Search and Indexing (Continued)**

Dragana Damjanovic, Claudia Plant, Sören Balko, Hans-Jörg Schek
User-Adaptable Browsing and Relevance Feedback in Image Databases

David Novak, Pavel Zezula
Indexing the Distance Using Chord: A Distributed Similarity Search Structure

Carlo Meghini, Nicolas Spyratos
Query Tuning through Refinement / Enlargement in a Formal Context

Wolfgang Nejdl, Raluca Paiu
I know I stored it somewhere - Contextual Information and Ranking on our Desktop

12:15	**Lunch**
14:00 – 14:30	**Invited Talk by *Keith Jeffery*: Digital Libraries in a Grid Environment**
14:30 – 15:30	**Working Groups**
15:30	**Coffee Break**
16:00 – 17:30	**Working Groups**
18:00	**Dinner**
20:00	**Gong Show**

Friday, April 1, 2005

9:00 – 10:30	**Presentations of Working Group Results**
10:30	**Coffee Break**
11:00 – 12:00	**Wrap-up Discussion**
12:15	**Lunch**
Departure	

Preliminary List of Participants

Maristella Agosti, Department of Information Engineering, University of Padua, agosti@dei.unipd.it

Sören Balko, ETH, sbalko@inf.ethz.ch

Stanislav Barton, Faculty of Informatics, Masaryk University of Brno, xbarton@fi.muni.cz

Matthias Bender, MPI, mbender@mpi-sb.mpg.de

Klaus Berberich, MPI, kberberi@mpi-sb.mpg.de

Ludger Bischofs, Software Engineering, University of Oldenburg, ludger.bischofs@informatik.uni-oldenburg.de

Peter Buneman, School of Informatics, University of Edinburgh, peter@cis.upenn.edu

Donatella Castelli, ISTI, donatella.castelli@isti.cnr.it

Sebastiano Colazzo, Polimi, colazzo@elet.polimi.it

Dragana Damjanovic, UMIT, dragana.damjanovic@umit.at

Giorgio Maria Di Nunzio, Department of Information Engineering, University of Padua, dinunzio@dei.unipd.it

Andrea Ernst-Gerlach, Institute of Informatics and Interactive Systems, University of Duisburg-Essen (for Norbert Fuhr), ernst@is.informatik.uni-duisburg.de

Nicola Ferro, Department of Information Engineering, University of Padua, nf76@dei.unipd.it

Peter Fischer, ETH (Kossmann), peter.fischer@inf.ethz.ch

Mike Freeston, UCSB, freeston@alexandria.ucsb.edu

Stefan Gradmann, Regionales Rechenzentrum, Universität Hamburg, stefan.gradmann@rrz.uni-hamburg.de

Fabian Groffen, CWI, Fabian.Groffen@cwi.nl

Wilhelm Hasselbring, Software Engineering, University of Oldenburg, hasselbring@informatik.uni-oldenburg.de

Yannis Ioannidis, University of Athens, yannis@di.uoa.gr

Keith Jeffery, CCLRC RAL, K.G.Jeffery@rl.ac.uk

Brian Kelly, UKOLN, University of Bath, B.Kelly@ukoln.ac.uk

Anastasios Kementsietsidis, University of Edinburgh, akements@inf.ed.ac.uk

Ioanna Koffina, ICS Forth, koffina@ics.forth.gr

George Kokkinidis, ICS Forth, kokkinid@ics.forth.gr

Manolis Koubarakis, Intelligence, TUC, manolis@intelligence.tuc.gr

Harald Krottmaier, Institute for Information Systems and Computer Media, University of Technology Graz, hkrott@iicm.edu

Steffanie Linde, ETH

Stefan Manegold, CWI, Stefan.Manegold@cwi.nl

Carlo Meghini, ISTI CNR, carlo.meghini@isti.cnr.it

David Novak, Faculty of Informatics, Masaryk University of Brno, xnovak@fi.muni.cz

Raluca Paiu, L3S, paiu@l3s.de

Paolo Paolini, Polimi, paolo.paolini@polimi.it

Frank Pentaris, UOA, frank@di.uoa.gr

Vito Perrone, Polimi, perrone@elet.polimi.it

Claudia Plant, UMIT, claudia.plant@umit.at

Hans-Jörg Schek, UMIT / ETH, hans-joerg.schek@umit.at

Heiko Schuldt, Information & Software Engineering, UMIT, heiko.schuldt@umit.at

Timos Sellis, National Technical University of Athens, timos@dblab.ece.ntua.gr

Michael Springmann, UMIT, michael.springmann@umit.at

Martin Theobald, MPI, martin.theobald@mpi-sb.mpg.de

Christos Tryfonopoulos, Intelligence, TUC, trifon@intelligence.tuc.gr

Michalis Vazirgiannis, Department of Informatics, Athens University of Economics & Business, mvazirg@aueb.gr

Timo Warns, Software Engineering, University of Oldenburg, timo.warns@informatik.uni-oldenburg.de

Gerhard Weikum, MPI, weikum@mpi-sb.mpg.de

Marcin Werla, Poznań Supercomputing and Networking Center, mwerla@man.poznan.pl

Manfred Wurz, Information & Software Engineering, UMIT, manfred.wurz@umit.at

Pavel Zezula, Faculty of Informatics, Masaryk University of Brno, zezula@fi.muni.cz

Christian Zimmer, MPI, czimmer@mpi-sb.mpg.de

Dynamically Making Use of Distributed Data Sources in a Grid Environment (Ext. Abstract)

Manfred Wurz and Heiko Schuldt

University for Health Sciences, Medical Informatics and Technology
Eduard-Wallnöfer–Zentrum 1   A–6060 Hall in Tyrol, Austria
[manfred.wurz|heiko.schuldt@umit.at]

Abstract. To avoid the cost of multiple and costly examinations, health care institutions are in need to share information about scientific insights and patient data more intensively and transparently. The need for seamless but still robust and secure collaboration is rising. Based on that scenario, this paper proposes an architecture for dynamically parallelizing service requests in a grid environment without the need to change existing and conscientiously tested functionality. The task of preparing software for parallel execution is split into an application-specific part of partitioning requests and re-integrating results and a generic component responsible for the actual parallel calls, state management, failure handling, and robustness.

Introduction

One major goal of grid computing is to establish highly flexible and robust environments to utilize distributed resources in an efficient and transparent way. Due to the highly dynamic nature of such environments where computational nodes may leave or join in, it is essential to bind service invocations to concrete service instances at run-time. This allows to flexibly react to changes in the environment. In a service-oriented world, application logic is encapsulated by means of services. Standards like SOAP over HTTP can be used for the invocation of (Web) services, and WSDL for accessing information on the capabilities of services. When several instances of the same service exist in a grid environment, then it should be possible to dynamically make use of as many service instances as possible by parallelizing a (Web) service call and by submitting requests in parallel to them.

The goal of this parallelization is twofold and depends on the characteristics of the services which are subject to parallelization. First, we aim to make use of as many services as possible (and therefore of the data accessible by those), to increase the quality of the result. This is particularly true for the access to data sources, encapsulated by dedicated services. Second, having multiple service instances accessible opens the possibility to speed up the processing of computationally intensive tasks. Whereas examples for the latter have been presented in detail in [6, 5] and large scale experiments with more than 2500 worker nodes have been demonstrated in [1] using MW class library [3], this work focuses on the goal of enhancing the quality of the request when accessing data sources by means of services.

The contribution of this work is to introduce an architecture of a service seeming to be an ordinary, callable service to the outside world, which is able to adopt its behavior controllable by optional quality of service criteria, and the resources available on a grid. In short, such *dynamic* services use meta information on the currently available service providers and their capabilities and partition the original request into a set of simpler requests of the same service types. These (sub-)requests are then submitted in parallel to as many service providers as reasonable, and their responses are finally integrated and returned as the result of the original request. In particular, this parallelization shall be carried out without the need to modify existing functionality or interfaces, and transparently to the user and developer. Due to its master/slave nature, it is especially suitable for grid environments [2].

To better illustrate the benefits of such an approach, the following scenario describes how these dynamic services can be used in healthcare applications.

A Sample Scenario: Genotype/Phenotype Correlation. *Donald, a patient, consults his family doctor telling him about pain in his chest, shortage of breath, and drowsiness. Besides these symptoms being an indicator for angina pectoris, Donald describes that starting a few hours ago, he additionally suffers from pain in his abdomen and changing paralysis of his right and left leg. Based on this description, the physician wants to ask for a second opinion and admits the patient to a hospital for further examination. The specialist who examines Donald in the hospital recognizes some very specific clinical artifacts which rise his interest (gigantism, a funnel chest, scoliosis, and acromacria). Due to these symptoms, Donald has to undergo clinical as well as molecular genetic examinations, since this clinical fingerprint might be caused through a fibrillin-1 (FBN1) gene mutation. To find out more about Donald's potential genetic mutation, his clinical fingerprint is used as an input for a genotype/phenotype correlation. The quality and statement of such a correlation is highly influenced by the amount of data that can be processed. Collaboration among various healthcare institutions, where data is made available by genotype/phenotype correlation services, is therefore of great importance. Luckily, the clinical fingerprint of Donald was indeed listed in the data set of a partnering hospital, connected to the grid and dynamically included in the search, and correlates with the Marfan–Syndrome. An expensive molecular biological examination in which this genetic defect is confirmed is conducted. Therefore, in addition to the acute vascular operation he has to undergo because of his clinical symptoms, his family can be invited for genetic screening to avoid similar costly, high risk operations in the future.*

Using the approach described in this work, the correlation service does not only consider the in-house service of a single hospital. Rather, it is a self-adaptive, *virtual* service that dynamically and in parallel calls all available genotype/phenotype correlation services in the system, thereby jointly accessing the data sources of several healthcare institutions. As a benefit of the architecture proposed, the client application as well as the genotype/phenotype correlation functionality does not have to be updated or rewritten to participate in or benefit from such a self adaptive environment.

Although the above example is taken from eHealth digital libraries, the proposed architecture can of course be applied to other domains as well. Whenever it is appropriate to dynamically replace a single invocation of a service by multiple invocations, dynamic adaptation and parallelization can be highly beneficial.

The remainder of this extended abstract is organized as follows. We briefly describe the overall architecture and show the benefits of dynamic parallelization for the application scenario introduced above. Finally, a conclusion and an outlook on future work is given.

Overall Architecture

The goal of the architecture we propose is to improve the usability of a single (Web) service, as well as to facilitate faster and less error prone development for grid environments. This approach is based on the observation that, following the current proliferation of service-oriented architectures, the number of services and service providers in a grid will significantly increase. Especially services which are provider independent and are not bound to special resources can be distributed fast and widely in a grid environment or be deployed numerously on demand. Although the availability of many congruent services as well as the computational resources thereby offered seem to be within reach, adapting functionality for parallel execution is still necessary and tedious.

The task of partitioning request parameters and re-integrating results afterwards is highly application-specific and, from our perspective, cannot be solved in a generic way. Although we see the potential to identify classes of applications according to the mechanism they partition and re-integrate requests which allows to have pre-built splitter and merger services, an expert in the problem domain will be necessary to tailor them for the specific need or perform some additional, application domain specific work. We name a service, enriched with the capability to partition incoming requests and reintegrate partial results, a *dynamic* service.

Apart from that we introduce a *Dynamizer* component, which can be built generically and which is responsible for state management, failure handling, and service discovery. While the latter characteristic is independent concerning the twofold approach described in the previous section, failure handling and state management differs. In case of a dynamic service aiming to leverage the result quality, the unexpected absence of a 'worker' service already in charge of a partial task does not hinder in producing a result. If partitioning the request of a computationally intensive task to gain better performance, the overall results rely on each sub task and the *Dynamizer* is responsible for compensating any failing 'worker' service.

As shown in Figure 1, the following logical units can be identified for dynamic services. The box in the center of the left side, labeled 'Payload Service', represents the actual service. It is responsible for providing application seman-

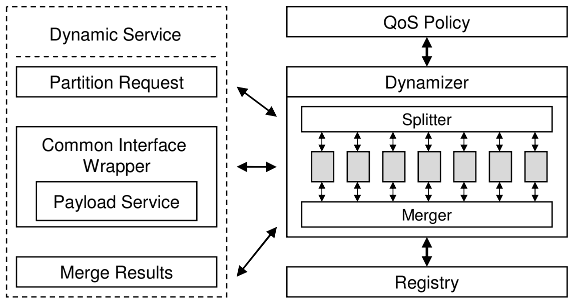


Fig. 1. Overall Architecture

tics, e.g., a genotype/phenotype correlation on a local database. This is usually a piece of business logic that has existed beforehand and is now supposed to be opened to the grid and enabled for parallel execution. To achieve this goal, it is surrounded by another box, labeled 'Common Interface Wrapper', which encapsulates the 'Payload Service' and enhances it with a common interface.

On top, 'Partition Request' encapsulates knowledge on how incoming parameters for the 'Payload Service' have to be partitioned, so that the original request can be decomposed into numerous new sub-requests. Each of these sub-requests can then be distributed on the grid and be processed by other instances of the originally targeted service. The box at the bottom ('Merge Results') integrates (partial) results returned from the grid to provide the original service requester with a consolidated result. It can therefore be seen as the counter operation to the 'Partition Request' service. The combination of these elements is referred to as 'Dynamic Service'.

To find instances of the originally targeted service (e.g., services where the description equals the one of the 'Payload Service'), a registry is used (depicted in the lower right corner of Figure 1) . This registry provides information on which services are available, how they can be accessed, and what their properties are (e.g., CPU load, connection bandwidth, access restrictions, etc).

The 'Dynamizer', depicted on the right hand side, makes use of the services mentioned above. It glues together the previously described services by making the parallel calls and coordinating incoming results. The 'Dynamizer' can interact with all services that adhere to a common interface, as ensured by the 'Common Interface Wrapper'. It can be integrated in environments able to call and orchestrate services, or it can be packaged and deployed together with specific services.

To make the best possible use of the 'Dynamizer', the user can send an optional description of the desired service quality along with the mandatory parameters needed to process the request. In this Quality of Service (QoS) policy, the user can, for example, describe whether the request should be optimized in terms of speed (select high performance nodes, and partition the input parameters accordingly), in terms of bandwidth (try to keep network usage low) or if it should aim for best accuracy (important for iterative approaches or database queries, where there is an option to use different data sources). Since these specifications can be contradictory, adding preferences to rank the user's requirements is of importance. To better illustrate the mechanisms within the 'Dynamizer' regarding the user specified QoS policy, we consider the following example: the specialist from the previously introduced healthcare scenario specifies that he wants to use as many genotype/phenotype correlation information as possible and as affordable within a 300 Euro budget. The 'Dynamizer' finds 7 services with a total of 2 gigabytes of searchable data, each charging 60 Euros per query. Alternatively, there are 40 services available provided by smaller institutions, having just searchable amounts of data starting from 4 megabytes up to 30 megabytes and charging .50 per query. The algorithms on how to reconcile the user specifications, the details of the QoS description language and how to integrate this best with our existing implementation is currently investigated.

Conclusion and Outlook

In this paper, we have stated the importance and the usefulness of an easy to use, straightforward to develop and robust architecture to dynamically parallelize (Web) service calls without the need to change existing functionality. In future work, we plan to implement the assignment of QoS policies to service requests as well as adding the ability to use semantically equivalent instead of congruent services. First experimental results will be refined and further empirical studies will be conducted to verify the validity of the approach described. When dealing with semantically equivalent services, (partial) results are likely to be heterogeneous, and mechanisms for integrating them have to be developed. This, additionally to defining appropriate metrics for semantic equivalence in the context described, is currently under investigation. Along with these changes, the fault tolerance, robustness, and scalability of the introduced 'Dynamizer' component is improved by integrating it with OSIRIS [4], a distributed workflow environment.

References

1. Kurt Anstreicher, Nathan Brixius, Jean-Pierre Goux, and Jeff Linderoth. Solving Large Quadratic Assignment Problems on Computational Grids. In *Mathematical Programming 91(3)*, pages 563–588, 2002.
2. Ian Foster and Carl Kesselman, editors. *The Grid 2, Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 2004.
3. Jean-Pierre Goux, Sanjeev Kulkarni, Jeff Linderoth, and Michael Yoder. An Enabling Framework for Master-Worker Applications on the Computational Grid. In *9th IEEE Int'l Symp. on High Performance Dist. Comp.*, pages 43–50, Los Alamitos, CA, 2000. IEE Computer Society Press.
4. C. Schuler, R. Weber, H. Schuldt, and H.-J. Schek. Scalable Peer-to-Peer Process Management - The OSIRIS Approach. In *Proceedings of the 2^{nd} International Conference on Web Services (ICWS'2004)*, pages 26–34, San Diego, CA, USA, July 2004. IEEE Computer Society.

5. M. Wurz, G. Brettlecker, and H. Schuldt. Data Stream Management and Digital Library Processes on Top of a Hyperdatabase and Grid Infrastructure. In *Pre-Proceedings of the 6th Thematic Workshop of the EU Network of Excellence DELOS: Digital Library Architectures - Peer-to-Peer, Grid, and Service-Orientation (DLA 2004)*, pages 37–48, Cagliari, Italy, June 2004. Edizioni Progetto Padova.
6. M. Wurz and H. Schuldt. Dynamic parallelization of grid–enabled web services. In *Proceedings of European Grid Conference 2005 (EGC 2005) (to appear)*, 2005.

Influence of Replication on Availability within P2P Systems

Timo Warns[1], Wilhelm Hasselbring[12], and Mark Roantree[3]

[1] Software Engineering Group, Carl von Ossietzky Universität Oldenburg, Germany,
`(timo.warns|hasselbring)@informatik.uni-oldenburg.de`
[2] OFFIS Research Institute, Germany,
[3] Interoperable Systems Group, Dublin City University, Ireland,
`mark@computing.dcu.ie`

Abstract. An increasing number of digital library management systems is developed following P2P architectures to overcome the bottlenecks of client/server architectures. Usually, the participating peers are less dependable than traditional servers. Hence, a P2P system needs to deal with failures of single peers to avoid overall system failures. Replication is a means to improve availability of resources. We empirically investigate the influence of replication techniques on availability by simulations. We focus on voting-based replication control strategies which offer one-copy-serializability in the context of our XPeer architecture.

1 Introduction

Peer-to-peer (P2P) digital libraries are highly dynamic as they shall facilitate data sharing among users. The churn rate is high as their peers enter and leave the system frequently. Hence, the dependability of peers is worse than the corresponding characteristics of traditional servers. Particularly, current P2P systems employ *small-world topologies*, *cross-partition pointers*, *self-organization* and *periodic description updates* to improve dependability [10]. In summary, these means aim at maintaining the topology and communication among available peers in the presence of failing peers.

P2P systems offer services which are delivered by participating peers. The operation of a service requires access to resources, like databases. Usually, these resources are hosted by the peers which deliver the service. If the hosting peers fail, the service crashes jointly. Hence, means of retaining communication and topology do not suffice to improve dependability of P2P services. Means must address the resources hosted on peers as well.

Replication and caching are suitable means to improve availability of resources [6]. Digital library systems provide data and metadata. Usually, caching is preferred to replication for data like audio and video files as this kind of content changes rarely. However, metadata is subject to replication, because it is updated by users frequently, e.g., to rate content. For example, Kovács et al. describe a peer-to-peer digital library which caches content and replicates metadata [7].

P2P systems differ from traditional distributed systems. For example, intermediate peers influence the dependability of communication, and fault characteristics of peers are worse than the corresponding attributes of traditional servers. Therefore, replication has to be investigated specifically for P2P contexts, because solutions for traditional distributed systems may not be appropriate. Our work addresses this issue for the voting-based replication strategies *Read-One-Write-All* (ROWA) and *Majority Consensus* [9] as a first step. Particularly, we explore the availability of read and write operations on replicated resources compared to non-replicated resources.

2 Related Work

Vanthournot et al. propose small-world topologies, self-organization, and cross-partition pointers as means of fault-tolerance within P2P systems [10]. They simulate P2P systems to investigate the resulting dependability. They focus on failures of peers and connections, but omit the issue of resources on peers.

Replication based on rumour spreading is proposed for P-Grid [4]. These algorithms ease consistency conditions to improve availability and performance. They focus on analyzing the communication overhead.

3 The XPeer architecture

P2P systems which are required to be highly dependable in the presence of frequent updates under a sequential consistency model come into question for voting-based replication. Our current work includes the XPeer architecture for data integration within such systems.

XPeer is a logical super-peer architecture which is well suited for digital libraries [1]. It addresses the issue of querying distributed data in a large scale context by realizing an integrated schema. This schema is formed from heterogenous information sources by classifying data sources into domains and creating user profiles for query optimizations. Information sources are integrated using XPeer's novel concept of super-peer application in a database environment. Super-peers are used to integrate information sources from clusters of interest or similarity. However, these sources are prone to disappear in a P2P architecture, causing problems for the query service and the optimisation process. The Replication Service described here is used to extend the original architecture.

4 Method

The availability of reading a replicated resource must be considered separately from the availability of writing, because the behaviour of replication control strategies may differ for reading and writing. We evaluated the resulting values by discrete event-simulation of scenarios. We developed a simulation model of peers following real-world implementations like Freenet [3] and incorporated techniques of replication.

A scenario consists of a P2P system with specific peers, connections, and their availabilities, a replication strategy and a distribution of replicas to peers. The set of all possible scenarios is infinite. We restrict ourselves to a subset of scenario classes for the investigation. The set of all scenarios can be characterized along five main dimensions: P2P architecture, P2P instantiation, faults, replication architecture, and replication deployment.

The P2P architecture describes the conceptual layout of a P2P system. The dimension is subdivided into degree of centralization, structure, and style of communication. The degree of centralization determines whether a system may have centralized elements, e.g., index servers. The topology of P2P systems may be structured, e.g., to a ring or small-world topology. The style of communication describes whether peers are able to communicate indirectly by relaying and forwarding messages. A P2P instantiation is a derivation of an actual P2P system from an architecture. It describes how many peers participate in the system and how they are connected. The faults dimension specifies the fault characteristics of peers and connections. We assume an exponential distribution of faults. For our purposes, the mean time to failure and the mean time to repair suffice, because the availability and reliability of peers and connections can be derived from these values [8]. The replication architecture describes the conceptual behaviour of replicas. Several classifications of replication are known [5, 12]. The replication instantiation specifies the actual number and distribution of replicas to peers.

5 Results

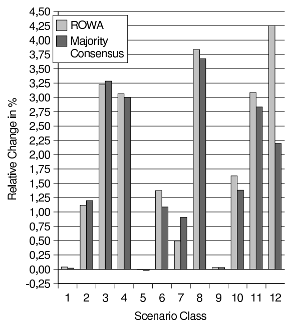
We derived 36 scenarios classes from the dimensions above. We chose decentralized and super-peer architectures with unstructured, mesh-like, and small-world topologies and indirect communication. The mesh-like topology is a special type of a structured topology, whereby each peer is connected to four neighbors to form a net. A small-world topology is characterized by short paths of intermediate peers between any pair of peers [11]. The number of peers is fixed to 49 for each scenario. The connections are chosen depending on the demands of the topology. The range of mean times to failure and mean times to repair is derived from observations of a real-world system [2]. Two simple types of weighted voting are considered for the replication architecture: Read-One-Write-All (ROWA) and Majority Consensus. The number of replicas is fixed to five for deployment. Their distribution to peers is managed in two ways: They are located on peers with best fault characteristics or are allocated according to a normal distribution. Each scenario class is simulated with a single read or a single write access to acquire the resulting availability. Additionally, each obtained P2P system is simulated with a non-replicated resource to be able to evaluate the relative influence of ROWA and Majority Consensus.

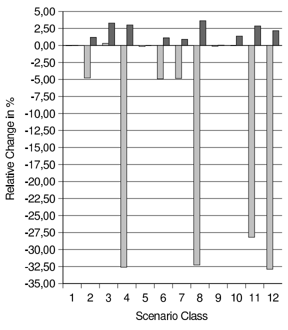
The results of the simulations for each scenario class are presented in table 1. The relative influence of the replication techniques compared to the correspond-

Table 1. Resulting Availability

Resulting Availability for Reading				**Replication Control Strategy**					
				None		*Read-One-Write-All*		*Majority Consensus*	
				Replica Distribution		**Replica Distribution**		**Replica Distribution**	
				Best Peers	*Normal Distr.*	*Best Peers*	*Normal Distr.*	*Best Peers*	*Normal Distr.*
Structure	*Unstructured*	**Fault Distribution**	*Uniform Good*	0.9993		0.9997		0.9995	
			Uniform Medium	0.9762		0.9871		0.9879	
			Normal Distr.	0.9470	0.9468	0.9775	0.9758	0.9781	0.9752
	Mesh-Like	**Fault Distribution**	*Uniform Good*	0.9998		0.9998		0.9996	
			Uniform Medium	0.9752		0.9886		0.9858	
			Normal Distr.	0.9692	0.9387	0.9740	0.9747	0.9780	0.9732
	Super-Peer	**Fault Distribution**	*Uniform Good*	0.9994		0.9997		0.9997	
			Uniform Medium	0.9628		0.9785		0.9761	
			Normal Distr.	0.9209	0.9099	0.9493	0.9486	0.9470	0.9299

Resulting Availability for Writing				**Replication Control Strategy**					
				None		*Read-One-Write-All*		*Majority Consensus*	
				Replica Distribution		**Replica Distribution**		**Replica Distribution**	
				Best Peers	*Normal Distr.*	*Best Peers*	*Normal Distr.*	*Best Peers*	*Normal Distr.*
Structure	*Unstructured*	**Fault Distribution**	*Uniform Good*	0.9992		0.9989		0.9995	
			Uniform Medium	0.9763		0.9296		0.9879	
			Normal Distr.	0.9470	0.9467	0.9499	0.6382	0.9781	0.9753
	Mesh-Like	**Fault Distribution**	*Uniform Good*	0.9996		0.9981		0.9998	
			Uniform Medium	0.9750		0.9271		0.9858	
			Normal Distr.	0.9690	0.9388	0.9220	0.6359	0.9778	0.9730
	Super-Peer	**Fault Distribution**	*Uniform Good*	0.9994		0.9983		0.9999	
			Uniform Medium	0.9627		0.9627		0.9760	
			Normal Distr.	0.9207	0.9098	0.6612	0.6106	0.9471	0.9297





Scenario Classes

1	Unstructured, Uniform Good
2	Unstructured, Uniform Medium
3	Unstructured, Norm. Distr., Best Peers
4	Unstructured, Norm. Distr., Norm. Distr.
5	Mesh, Uniform Good
6	Mesh, Uniform Medium
7	Mesh, Norm. Distr., Best Peers
8	Mesh, Norm. Distr., Norm. Distr.
9	Super Peer, Uniform Good
10	Super-Peer, Uniform Medium
11	Super-Peer, Norm. Distr., Best Peers
12	Super-Peer, Norm. Distr., Norm. Distr.

Fig. 1. Relative Change Reading / Writing

ing scenario classes without replication is illustrated in figure 1. The ROWA strategy requires access to a single replica for reading. We expected that the strategy heavily improves the availability of reading. It has its best influence for availability of reading for scenario class 12 with about 4.25%. The worst influence occurs for scenario class 5 where no influence was measured at all. Not surprisingly, it improves reading for most scenarios. However, for some scenario classes $(1, 5, 9)$ the influence is negligible.

The ROWA strategy requires access to all replicas for writing. Hence, all hosting peers must be available for executing a write operation successfully. We expected that the strategy heavily decreases availability of writing, which was confirmed with relative influences ranging from -32.89% to 0.15%.

The Majority Consensus strategy requires access to more than half of the replicas for reading and writing. We expected that the strategy improves both availabilities. The influence for reading was expected to be worse than the influence of ROWA, because access to more than one replica is required. It has its best influence for scenario class 8 with a relative improvement of about 3.68%. The worst influence occurs for class 5 where its influence is negligible. In general our expectation was confirmed as the strategy improves availability for reading and writing. The influence for read operations was worse than the influence of

ROWA. However, it exceeds ROWA for the scenario classes 2, 3, and 7. It is interesting to see that it does not decrease availability significantly for any scenario class we chose.

A broader generalization of the simulation results is a topic for future work. Even small changes to the scenario may have high impact on the resulting values. However, our results already indicate at this stage that voting-based replication strategies are a favourable replication technique for P2P systems when high consistency is required. Even if the peers of the scenario classes had fault characteristics worse than traditional servers, Majority Consensus improves availability of reading and writing.

References

1. Z. Bellahsène and M. Roantree. Querying distributed data in a super-peer based architecture. In F. Galindo, M. Takizawa, and R. Traunmüller, editors, *Database and Expert Systems Applications, 15th International Conference, DEXA 2004*, volume 3180 of *Lecture Notes in Computer Science*, pages 296–305. Springer, 2004.
2. R. Bhagwan, S. Savage, and G. M. Voelker. Understanding availability. In F. Kaashoek and I. Stoica, editors, *Peer-to-Peer Systems II, Second International Workshop*, volume 2735 of *Lecture Notes in Computer Science*, pages 256–267. Springer, 2003.
3. I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In H. Federrath, editor, *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 / 2001 of *Lecture Notes in Computer Science*, pages 46–66. Springer, July 2000.
4. A. Datta, M. Hauswirth, and K. Aberer. Updates in highly unreliable, replicated peer-to-peer systems. In *23rd International Conference on Distributed Computing Systems (ICDCS 2003)*, pages 76–87. IEEE Computer Society, 2003.
5. S. B. Davidson, H. Garcia-Molina, and D. Skeen. Consistency in partitioned networks. *ACM Comput. Surv.*, 17(3):341–370, 1985.
6. W. Hasselbring. Federated integration of replicated information within hospitals. *International Journal on Digital Libraries*, 1(3):192–208, Nov. 1997.
7. L. Kovács, A. Micsik, M. Pataki, and R. Stachel. Collaboration of loosely coupled repositories using peer-to-peer paradigm. In M. Agosti, H.-J. Schek, and C. Türker, editors, *DELOS Workshop: Digital Library Architectures*, pages 85–92. Edizioni Libreria Progetto, Padova, 2004.
8. M. R. Lyu, editor. *Handbook of Software Reliability Engineering.* IEEE Computer Society Press and McGraw-Hill Book Company, 1996.
9. R. H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Trans. Database Syst.*, 4(2):180–209, 1979.
10. K. Vanthournot and G. Deconinck. Building dependable peer-to-peer systems. In *DSN 2004 Workshop on Architecting Dependable Systems*, 2004.
11. D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, June 1998.
12. M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso. Database replication techniques: a three parameter classification. In *Proceedings of 19th IEEE Symposium on Reliable Distributed Systems (SRDS 2000)*, pages 206–217. IEEE Computer Society, 2000.

Adaptive replication strategies and software architectures for peer-to-peer systems

Ludger Bischofs[1,2], Simon Giesecke[1], Wilhelm Hasselbring[1,2], Heiko Niemann[1], and Ulrike Steffens[2]

[1] Carl von Ossietzky University of Oldenburg, Software Engineering Group, 26111 Oldenburg (Oldb.), Germany, {*giesecke,hasselbring,niemann*}@informatik.uni-oldenburg.de

[2] Kuratorium OFFIS e.V., Escherweg 2, 26121 Oldenburg (Oldb.), Germany, {*ludger.bischofs,ulrike.steffens*}@offis.de

1 Introduction

The use of replication techniques for data in distributed information systems aims at improving non-functional properties of these systems. Particularly, availability, reliability, and performance should be increased. When considering heterogeneous, autonomy-preserving systems such as distributed digital libraries, the replication techniques employed should be able to adapt to varying properties of the individual systems at run-time.

Adaptive replication strategies need to be realized by a software architecture which provides the context for their implementation. Digital library systems whose information providers are organizationally closely related, for example under the umbrella of a single institution, should be tightly coupled, in order to allow for a high degree of consistency to be maintained. Other digital libraries, such as those of separate publishers, should be coupled less tightly, to allow for retaining their autonomy. This can be realized by a replication strategy based on peer-to-peer (P2P) techniques which combines intra-institutional and inter-institutional replication strategies. The overall replication strategy is evaluated and optimized at run-time, such that domain-specific requirements are incorporated into a multi-dimensional problem model.

2 Data replication strategies

Data *replication* aims at increasing availability, reliability and performance of data accesses by storing data redundantly [1–3]. A copy of a replicated data object is called a *replica*. Replication ensures that all replicas of one data object are automatically updated when one of its replicas is modified. Replication involves conflicting goals with respect to guaranteeing consistency, availability and performance [4]. An improvement of one of these properties usually implies a degradation of the others.

Replication in a distributed system is realized by a *replication strategy*, which is often controlled by *replication managers*. The quality of a strategy can be measured by the *correctness criterion* it safeguards. The strictest criterion is that of *one-copy-serializability*: The concurrent execution of a set of distributed transactions is one-copy serializable, if its effect is equivalent to the sequential execution of the transactions on a non-replicated database [5]. Besides this criterion a number of other correctness criteria have been defined, which are less strict and thus easier to fulfil.

Replication can be performed in an *eager* or *lazy* way [6]. In the case of eager replication, when one replica is modified by a transaction, the other replicas of the concerned data object are updated within the original database transaction, as opposed to lazy replication where only the originally accessed replica is updated within the original transaction, while the other replicas are updated in separate transactions. The classification into *synchronous* and *asynchronous* replication strategies is essentially equivalent to this classification. Combinations of synchronous and asynchronous replication have also been studied [7, 8].

The concept of an adaptive replication manager was first proposed in [9]. Its goal is to achieve an optimal trade-off between tight and loose coupling of information systems [10], which is realized by dynamically switching participating systems between being synchronously and asynchronously updated.

3 Peer-to-peer systems

Peer-to-peer systems are distributed systems that follow a communication model of equal nodes (peers) which communicate directly with each other. In the following we introduce different styles of peer-to-peer architectures before we describe the specific benefits of replication within peer-to-peer architectures.

3.1 Styles of peer-to-peer architectures

The class of peer-to-peer architectures can roughly be divided into pure and hybrid peer-to-peer architectures. *Pure peer-to-peer architectures* are completely decentralized. Each peer in the network is equipped with both client and server functionality and the architecture does not contain any central servers at all.

Hybrid peer-to-peer architectures combine characteristics of pure peer-to-peer architectures and client/server architectures. Some services are offered by servers and are therefore centralized, whereas other services are based on the peer-to-peer communication model. Hybrid architectures can further be subdivided into *centralized architectures* and *super-peer architectures.* In centralized architectures there are nodes with pure server functionality which offer exclusive services. Super-peer architectures are also called *hierarchical architectures* and combine the concepts of decentralized and centralized architectures. A super-peer is a peer which acts as a server for a set of ordinary peers and usually interacts with other super-peers. Ordinary peers are typically organized in clusters together with a single super-peer. Super-peer architectures are particularly suited to support both intra- and inter-institutional cooperation because they allow for representing the structures of an institution within the super-peers' clusters as well as the structures among different institutions by connecting the different super-peers accordingly.

3.2 Peer-to-peer replication

Today's replication strategies often depend on centralized solutions. Replicating heterogeneous, autonomy-preserving information systems, however, requires flexible and well-coordinated approaches. Intra- and inter-institutional replication should be addressed by an integrated approach, where institutions are coupled via a suitable peer-to-peer infrastructure.

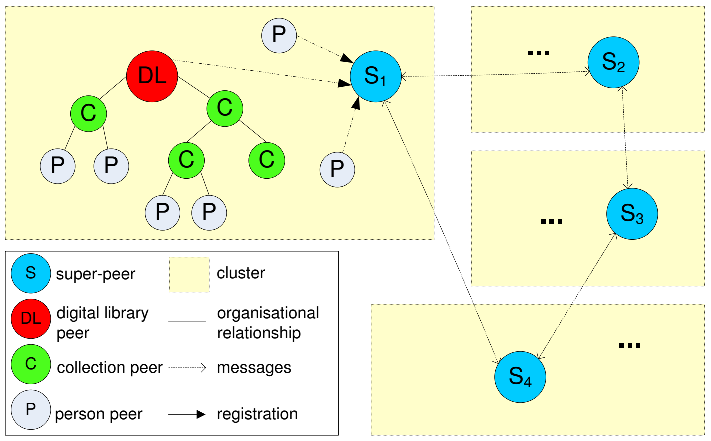


Fig. 1. Organization-oriented super-peer architecture for digital libraries

In figure 1, a super-peer architecture reflecting the requirements of different cooperative digital library systems is shown. We have chosen a super-peer approach because it allows to structure the logical network in accordance with the organizational structure. This architecture is one example of how to model specific organizational requirements by extending peer-to-peer architectures by an additional layer which provides information on the organizational context (cf. [11–13]). The architecture forms a basis for intra- and inter-institutional integration and replication of distributed data resources and services. Digital libraries together with specialized subcollections as well as individual users are represented by different types of peers. The connections among these peers as shown in the upper right of figure 1 are only one example of how a library may be organizationally structured. Other, non-hierarchical structures are also supported. Documents and also services may be controlled by the digital library as superordinate institution, by single collections and also by single person peers who e.g. can offer their individual metadata or link lists to interesting web documents.

The intra-institutional structure as described above is complemented by an inter-institutional coupling with the help of super-peers. The super-peers impose a partition into clusters of organizationally closely related peers, where one cluster may comprise one or more institutions. Super-peers have more responsibilities than ordinary peers. A super-peer stores metadata concerning the structure of the institutions under its auspices and of the data resources and services it provides. Thus, its main task is to mediate between the different clusters.

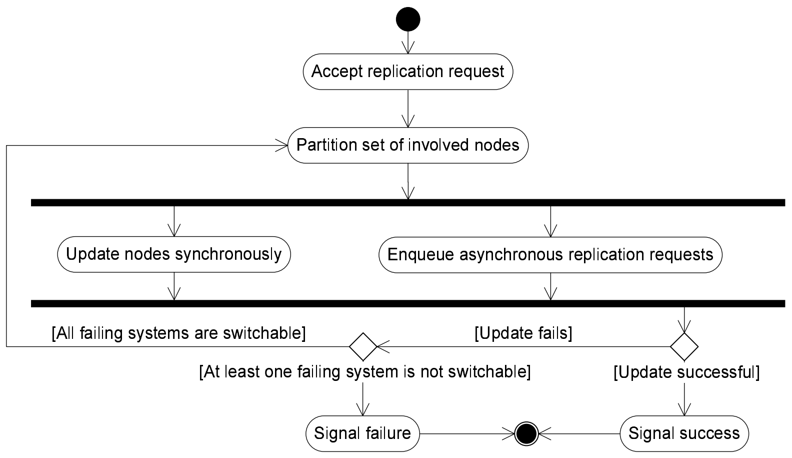

Accept replication request
Partition set of involved nodes
Update nodes synchronously
Enqueue asynchronous replication requests
[All failing systems are switchable]
[Update fails]
[At least one failing system is not switchable]
[Update successful]
Signal failure
Signal success

Fig. 2. Adaptive replication strategy

The structure given by an organization-oriented super-peer network can serve as foundation for the realization of adaptive intra- and inter-institutional replication strategies. Within a single institution the overall information system is usually composed of multiple component information systems, which must replicate data permanently due to high availability requirements. An advanced replication strategy is required to facilitate both high consistency and autonomy of the component information systems, each of which must not be vitally impaired when other systems fail. An optimal trade-off between these conflicting requirements can be reached by combining synchronous and asynchronous replication into an *adaptive replication strategy.* Each system may be replicated either synchronously or asynchronously at any time. Switching between synchronous and asynchronous replication should be configurable and adaptive with respect to the current network configuration status. The configuration is performed using a system of rules, which is continuously evaluated at run-time to ensure adaptivity. The correctness criterion of the replication strategy is evaluated by means of an appropriate consistency measure, i.e. a measure of the probability of consistent accesses.

A transaction that modifies a replicated data object is called a replication request. It is initiated by any node connected to the peer-to-peer network and is sent to its replication manager: One possibility to determine the replication manager is to use the digital library peer as intra-institutional replication manager, if the initiating node is contained within an institution (cf. 1). If the replication request is not confined to the originating institution, or if the initiating peer is not assigned to an institutional digital library peer, its super peer is used as an inter-institutional replication manager. In figure 2, the process realizing an adaptive replication strategy is illustrated for one replication request. By evaluating the system of rules, the involved nodes are split into two groups, those that are to be updated synchronously and asynchronously, respectively. After that, both groups of nodes are processed. The nodes in the synchronous group are directly updated, while the update requests for the nodes in the asynchronous group are enqueued into a message queue. All node updates are performed in parallel within one transaction. If all synchronous updates were successful, the processing of the replication request is completed and positively acknowledged. If a synchronous update fails, the replication manager checks whether the failing systems may be switched to asynchronous update mode. If this is the case, the corresponding systems are switched and the processing of the replication request is restarted. If some system could not be switched, the replication request fails.

Asynchronous replication requests in the message queue are continuously processed in an independent thread of execution. If a system to be updated is not available, the corresponding request stays in the queue until the update has successfully been performed.

4 Peer-to-peer replication for digital libraries

Today's digital library systems cooperate in manifold ways. They exhibit a varying internal organizational structure, e.g. given by the introduction of specialized subcollections or by extraction of project-specific reference libraries [14]. Furthermore, different business models also including the ability to count the cost of library access have to be taken into account [15]. Against this background, the ability to map intra- and inter-institutional requirements to the underlying digital library system as presented above is of paramount importance. Libraries including resources other than traditional library documents like e.g. resource collections in e-science [16] or health information systems [10, 11, 13, 17] emerge. This calls for the determination of detailed strategies not only for searching these collections [18] but also for replication of both resources and services as claimed in this paper.

References

1. Bernstein, P.A., Goodman, N.: The failure and recovery problem for replicated databases. In: Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing, New York, NY, ACM Press (1983) 114–122
2. Mustafa, M., Nathrah, B., Suzuri, M., Osman, M.A.: Improving data availability using hybrid replication technique in peer-to-peer environments. In: Proc. 18th International Conference on Advanced Information Networking and Applications (AINA'04), IEEE CS Press (2004) 593–598

3. Loukopoulos, T., Ahmad, I.: Static and adaptive data replication algorithms for fast information access in large distributed systems. In: Proc. 20th International Conference on Distributed Computing Systems (ICDCS 2000), IEEE CS Press (2000) 385–392
4. Hasselbring, W.: Federated integration of replicated information within hospitals. International Journal on Digital Libraries **1** (1997) 192–208
5. Traiger, I.L., Gray, J., Galthier, C.A., Lindsay, B.G.: Transactions and consistency in distributed database systems. ACM Transactions on Database Systems **7** (1982) 323–342
6. Gray, J., Helland, P., O'Neil, P., Shasha, D.: The dangers of replication and a solution. SIGMOD Rec. **25** (1996) 173–182 Proceedings of the SIGMOD 1996 conference.
7. Lubinski, A., Heuer, A.: Configured replication for mobile applications. In Barzdins, J., Caplinskas, A., eds.: Databases and information systems, Dordrecht, Niederlande, Kluwer Academic Publishers (2001) 139–151
8. Röhm, U., Böhm, K., Schek, H.J., Schuldt, H.: FAS – A Freshness-Sensitive Coordination Middleware for a Cluster of OLAP Components. In: 28th International Conference on Very Large Data Bases (VLDB 2002). (2002) 754–765
9. Niemann, H., Hasselbring, W., Hülsmann, M., Theel, O.: Realisierung eines adaptiven Replikationsmanagers auf Basis der J2EE-Technologie. In: Datenbanksysteme für Business, Technologie und Web (BTW). Volume 26 of GI-Edition – Lecture Notes in Informatics., Bonn, Bonner Köllen Verlag (2003) 443–452
10. Niemann, H., Hasselbring, W., Wendt, T., Winter, A., Meierhofer, M.: Kopplungsstrategien für Anwendungssysteme im Krankenhaus. Wirtschaftsinformatik **44** (2002) 425–434
11. Bischofs, L., Hasselbring, W., Schlegelmilch, J., Steffens, U.: A hierarchical super peer network for distributed artifacts. In Agosti, M., Schek, H.J., Türker, C., eds.: DELOS Workshop: Digital Library Architectures. (2004) 105–114
12. Bischofs, L., Hasselbring, W.: A hierarchical super peer network for distributed software development. In: Proc. Workshop on Cooperative Support for Distributed Software Engineering Processes (CSSE 04). (2004) 99–106
13. Bischofs, L., Hasselbring, W., Niemann, H., Schuldt, H., Wurz, M.: Verteilte Architekturen zur intra- und inter-institutionellen Integration von Patientendaten. In: Tagungsband der 49. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (GMDS 2004). (2004) 87–89
14. Schmidt, J.W., Schröder, G., Niederée, C., Matthes, F.: Linguistic and Architectural Requirements for Personalized Digital Libraries. International Journal of Digital Libraries **1** (1997) 89–104
15. Weber, R.: Chablis - Market Analysis of Digital Payment Systems. Institutsbericht, Technische Universität München, Institut für Informatik (1998)
16. Frommholz, I., Knezevic, P., Mehta, B., Niederée, C., Risse, T., Thiel, U.: Supporting Information Access in Next Generation Digital Library. In: Sixth International Workshop on Digital Library Architectures, Cagliari, Italy, Edizioni Progetto Padova (2004) 49–60
17. Wurz, M., Brettlecker, G., Schuldt, H.: Data stream management and digital library processes on top of a hyperdatabase and grid infrastructure. In: DELOS Workshop: Digital Library Architectures. (2004) 37–48
18. Klas, C.P., Fuhr, N., Schaefer, A.: Evaluating Strategic Support for Information Access in the DAFFODIL System. In: Research and Advanced Technology for Digital Libraries. Proc. European Conference on Digital Libraries (ECDL 2004). Volume 3232 of Lecture Notes in Computer Science., Springer (2004) 476–487

Distributed Services Architecture in dLibra Digital Library Framework

Cezary Mazurek and Marcin Werla

Poznan Supercomputing and Networking Center
Poznan, Poland
{mazurek,mwerla}@man.poznan.pl

Abstract. Architecture is one of the key factors influencing all distributed system components. It often decides about overall functionality, performance and flexibility. In this article we intend to describe the design of the first Polish digital library framework called dLibra, which has been developed in PSNC since 1999. We show how architecture based on modular and distributed services can be used to split digital library functionality. Such division gives opportunities for improving services availability and expansion possibilities. We also want to present mechanisms that we use to assure stable and continuous work of our distributed digital library framework, making it independent of various negative circumstances.

1 Introduction

The dLibra Digital Library Framework [1, 2] has been developed in Poznan Supercomputing and Networking Center since 1999. The first dLibra-based digital library (DL) was the Digital Library of the Wielkopolska Region (WBC) [3]. It was started in October 2002 and now it consists of over 3000 various publications grouped into four thematic collections: cultural heritage, regional materials, educational materials and music notes. Such number of publications makes from WBC one of the largest Polish digital libraries. In the end of November 2004 the second dLibra-based digital library was deployed – the Wroclaw University of Technology Digital Library (BCPWr) [4]. There are also four other test dLibra installations academic libraries, and in the nearest future three new regional digital libraries will be started.

Due to the diversity of mentioned digital libraries, many different aspects must be taken into account during the dLibra development. Each DL has its own specific publications – for example the majority of WBC publications are relicts of writing and old documents associated with social life of the Wielkopolska Region. Such resources are mostly stored in a graphical form, in formats like DjVu, PDF or JPG scans. All that publications consist of quite large files and their content is often not searchable. On the other hand, there are academic DL systems, like one of test dLibra installations in AGH University of Science and Technology, where a typical publication is an academic script stored as a set of HTML pages or small PDF file with searchable text content. Such differences requires support in many areas – from format dependent publication structure analysis during the publication upload process, to sophisticated mechanisms for content indexing and searching, to different publication view and download possibilities. Another important element is the amount of stored publications. When there are many gathered resources, and a large number of readers accessing it, the overall system performance becomes a crucial parameter.

In the following sections we want to show the way in which we designed the internal dLibra architecture and its basic mechanisms to create an efficient, flexible, distributed and error-resistant digital library system. The next section describes the structure of distributed services architecture developed in the dLibra Digital Library Framework. We also show an overview of these services functionality. The third section is focused on mechanisms for improving services reliability and availability in the dLibra framework. We try to demonstrate our approach to improving the dLibra stability in some extreme situations, like very heavy user requests load or network communication errors. Those mechanisms are an integral part of dLibra services and service management system. In the last section we point out some directions of our future, digital library-related works.

2 dLibra Architecture

The initial dLibra architecture and design was based on experiences from previous PSNC projects. We assumed that the dLibra environment should consist of a number of portable, distributed services. Portability was achieved by choosing Java™ as the programming language. Further works and practice from dLibra deployments formed the current dLibra structure. This structure is based on a set of cooperating remote services. All these services together create the complete dLibra-based digital library. Each service can be started on a separate computer, but they can also be connected into service groups running on the same machine. When services are started on different hosts, they use Java RMI technology to communicate with each other [5]. Six of dLibra services give together the entire dLibra server functionality. These services are:

– The Metadata Server – gives a possibility to define, modify and remove metadata attributes that are used to describe digital library publications. It also gives access to dictionaries and thesauri with values of all attributes. It is responsible for managing digital library directories and collections. In addition, it allows adding, modifying and deleting publications, and it has possibilities to manage lists of languages defined in the DL system. Moreover, it has a module for performing periodic metadata consistency test.
– The Content Server – gives access to all gathered digital content. Before sending content to the client, this service is able to compress it or encrypt and send securely. The Content Server is also used to store the publications content. Resuming is supported during both publication upload and download.
– The Search server – allows users to search through all gathered content and metadata. It also contains indexing functionality, which prepares indexes used during search.
– The Distributed Search Server – is used to harvest remote dLibra instances by means of the OAI-PMH [6] protocol. It also gives the user a possibility to search

through gathered remote metadata. In fact, any OAI-PMH-enabled repository can be harvested and searched using that service.

– The User Server – contains all user-related data and allows users authentication and authorization. It is also used to create groups of users and to grant users different digital library rights, from library administration to simple publication view.

As we mentioned before, all the above services give together the entire dLibra digital library functionality. However, at least two more elements are required to create a fully functional system. There must be a possibility to connect all these services and create an entry point to the system for both external applications and users.

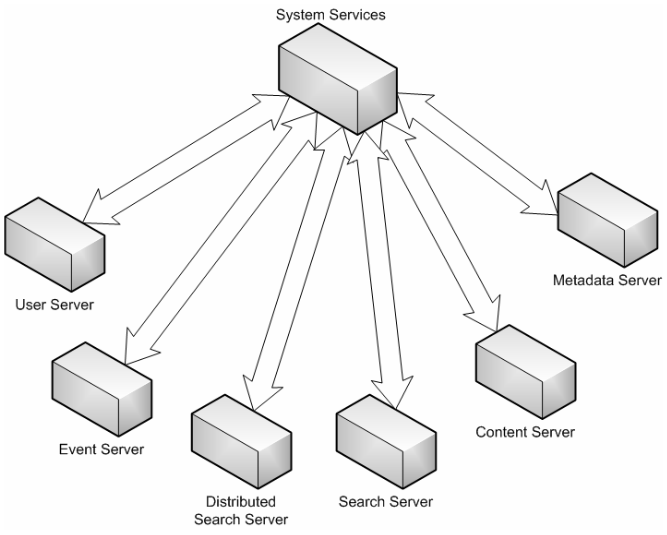


Fig. 1. Distributed dLibra services architecture

Connection between services is achieved with two additional services (see Figure 1). The first of them is a service called the System Services. It can be treated as a broker of services for single instance of the distributed digital library. It allows inter service communication and handles services addresses resolving, connecting and authorization. For example, when the Search Server wants to refresh its indexes, it asks the System Services for the Content Server and the Metadata Server. The System Service checks if such services are registered, if they are available and if the Search Server is authorized to access them. If all those conditions are met, as a response the Search Server receives references to the requested services. In order to become available to other services, each service must register itself in the chosen System Services. Services registered in one System Services create a digital library.

The second additional system level service is the Event Server. It allows services to communicate with the event messaging system. It is very useful when one service wants to notify some other services about a particular event. A good example of this mechanism can again be a process of refreshing search indexes (see Figure 2). Just after start, the Search Server registers in the Event Server for events related with the modification of gathered content and metadata. When a new publication is added, modified or removed, the Metadata Server sends an event notification to the Event Server. Next, the Event Server forwards this event to all services registered for this event type. After receiving such event, the Search Server can decide if index refreshing is required or maybe just some data should be removed from the index.

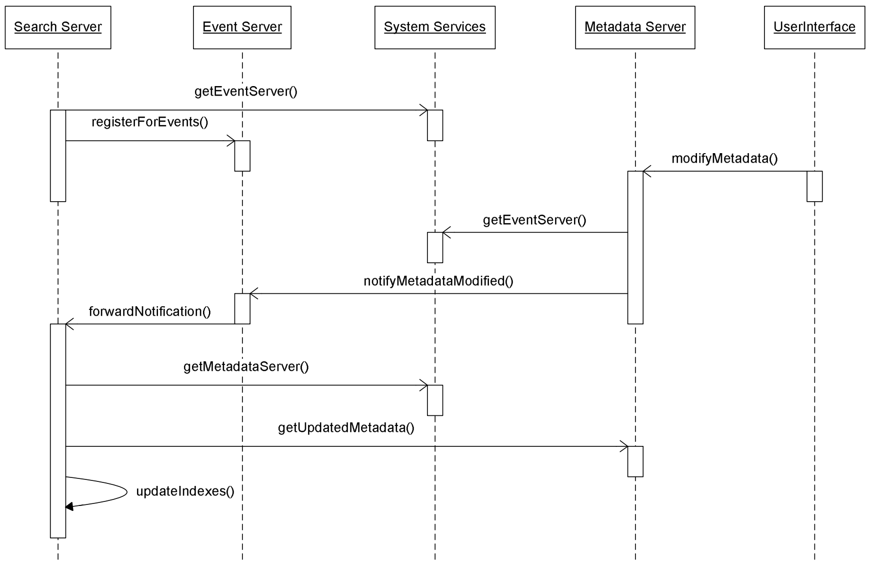


Fig.2. Sequence diagram describing Event Server event passing

In Figure 2 there is an element called User Interface. This corresponds to two additional parts of the dLibra Framework. One of them is WWW Service and the second is the Editor/Administrator application. The WWW Service is designed as a read only entry point to the system. It can be used by readers to access gathered resources. Content browse and searching are the main functionality of this service, but it is also an OAI-PMH data provider, and it has many user-friendly features like RSS [7] feeds with information newly added publications, publications ranking etc. This functionality is realized with the use of all other dLibra services reached through the System Services. The Editor/Administrator Application is an application for librarians and library administrators. It allows adding and modifying library resources and managing all library items.

3 Improving services availability in dLibra framework

The distributed architecture of the described dLibra services requires additional mechanisms for improving system reliability and availability. When one of the services stops responding, the library may become less functional (for example in case of the Search Server failure) or may not be functional at all - when the User Server or Metadata Server fails. To prevent such situations, a number of mechanisms were introduced.

The first of them is the way of service resolving done by the System Services. There is a possibility to create such services configuration, in which multiple instances of the most crucial services are started. Before the System Services gives one service access to another service, it tests the requested service functionality. When the tested instance of a given service fails, the System Services can return reference to other instance. Such instance switch is transparent to other services. With addition of services load monitoring functionality, this mechanism can also be used for load balancing between service instances.

The second mechanism is internal services monitoring. Each service is periodically checked if it is responding or has enough processor time for its tasks. This check is performed by a special service wrapper based on an open-source Java ServiceWrapper project [8]. The Wrapper can restart or shutdown the service when, for some reason, it stops responding or when host processors are overloaded for a longer period of time (for example during DoS attacks [9]). This service monitoring is done locally so it is independent of the state of network connections.

Another reliability improving mechanism is implemented in events sending and receiving parts of the dLibra framework. When service generates an event, it is not directly sent to the Event Server, but it is stored in a persistent storage. This storage is implemented with Hibernate [10], so it can be based on many types of relational databases. All stored events are read by a specialized Event Sender thread. This thread tries to send events to the Event Server. If connection to the Event Server is lost, all events stay in the storage until there is a possibility to send them again. On the other hand, when the Event Server retrieves an event, it also stores the event before trying to send it to registered services. Each service, while registering for events, gives the Event Server special timeout parameter. This parameter describes how long the events should be stored in the Event Server, if the registered service becomes unavailable. If the registered service becomes available again, all events stored for this service will be passed to it.

4 Future works

We think that next dLibra development stages will bring this distributed digital library framework closer to grid technologies. In order to do so, it will be necessary to extend our services model. Each service should gain the ability of describing itself with metadata. On top of the System Services there must be some kind of a new, much more advanced service – a dynamic distributed digital library services broker.

This should allow automated service discovery and creation of virtual DL organizations. Such active organizations of services could be used to create distributed digital collections from resources gathered in heterogeneous DL systems. We can also imagine Information Retrieval Grid services based on different distributed digital libraries [11, 12]. By creating an environment for advanced cooperation of computational grid services, grid data management systems and digital libraries we want to give an opportunity for advanced usage of digital libraries in sophisticated grid scenarios [13].

References

[1] Gruszczyński, P.; Mazurek, C.; Osinski S.; Swedrzynski A.; Szuber S. "dLibra Content Maintenance for Digital Libraries" in *Euromedia'2002*, pages 28–32, 7*th* Annual Scientific Conference, April 2002.

[2] Mazurek C.; Stroiński M.; Swędrzyński A.; „dLibra – Integrated Framework for Publishers and Libraries" – poster at 7th European Conference on Digital Libraries, Torndheim, Norway, August 2003.

[3] Digital Library of Wielkopolska Region. http://www.wbc.poznan.pl/.

[4] Wroclaw University of Technology Digital Library. http://dlib.bg.pwr.wroc.pl/.

[5] Hicks, M.; Jagannathan, S.; Kesley, R.; Moore, J.-T.; Ungureanu, C. "Transparent Communication for Distributed Objects in Java". ACM Java Grande Conference, pages 160-170, June 1999.

[6] Lagoze, C.; Van de Sompel, H. – "The Open Archives Initiative: Building a low-barrier interoperability framework", pages 54-62, Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries, Roanoke, VA, USA, June 2001.

[7] Hammersley , B. "Content Syndication with RSS". O'Reilly. 1st Edition. March 2003.

[8] Mortenson, L. "What is the Java Service Wrapper?". http://wrapper.tanukisoftware.org/doc/english/introduction.html

[9] CERT Coordination Center. "Denial of Service Attacks" http://www.cert.org/tech_tips/denial_of_service.html

[10] Cengija, D. "Hibernate Your Data". O'Reilly ONJava. 2004. http://www.onjava.com/pub/a/onjava/2004/01/14/hibernate.html

[11] Larson, R. R. "Distributed IR for Digital Libraries" in LNCS 2769, p. 487 – 498, 7th European Conference on Digital Libraries, Torndheim, Norway, August 2003.

[12] Dovey, M. J.; Gamiel, K.; "GRID IR — GRID Information Retrieval". Poster at EuroWeb 2002. Accessed from http://www.gridir.org/

[13] Kosiedowski, M.; Mazurek, C; Werla, M. – „Digital Library Grid Scenarios" in European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology, 25-26.05.2004, London, U.K. Workshop Proceedings, p. 189 – 196.

Integrating XML Data Sources using RDF/S Schemas: The ICS-FORTH Semantic Web Integration Middleware (SWIM)

Ioanna Koffina[1], Giorgos Serfiotis[1], Vassilis Christophides[1], Val Tannen[2], and Alin Deutsch[3]

[1] Institute of Computer Science, FORTH
Vassilika Vouton P.O 1385 GR 71110, Heraklion, Greece
and
Department of Computer Science, University of Crete
GR 71409, Heraklion, Greece
{koffina,serfioti,christop}@ics.forth.gr

[2] Computer and Information Science Department, UPenn
200 South 33rd Street, Philadelphia, Pennsylvania, USA
val@cis.upenn.edu

[3] Department of Computer Science & Engineering, UCSD
9500 Gilman Drive La Jolla, CA 92093, USA
deutsch@cs.ucsd.edu

1 Introduction

Digital libraries are collections of resource descriptions (also called metadata) that actually describe the catalogue data (i.e., digitized information). In general, these metadata are stored in diverse sources (e.g., relational or object databases, XML data sources, text files) distributed on the Web. One of the main challenges for the digital library community is the integration of such metadata sources in order to provide users with a common vocabulary for searching and browsing them.

The Semantic Web (SW) offers relevant approaches and standards that can handle these problems. More precisely, RDF (or other SW ontology languages) can be used as a common framework for expressing the information by providing a semantically rich representation language for metadata. In this context, a SW integration middleware (SWIM) should be employed for the integration of the heterogeneous and distributed sources. In particular, we propose a SWIM that provides a useful, comprehensive and high-level access to library metadata that reside in relational databases (RDB) or XML sources, by offering a virtual, mediated RDF/S schema.

There are many issues involved in the functionality of a SWIM. In particular, a SWIM should facilitate users to formulate queries against the mediated RDF/S schema using declarative languages (such as RQL [7]), as well as, support further abstraction levels using declarative view definition languages (e.g., RVL [10]). In a nutshell, SWIM should offer the following services: (a) establish mapping rules between XML and RDF and between RDB and RDF, (b) verify the conformance of these mappings w.r.t the semantics of the employed schemas, (c) reformulate RDF/S queries against RDB or XML sources, and (d) combine these queries with RVL views.

In order to address effectively and efficiently the above requirements, we should choose a uniform and expressive logic framework to define SWIM integration services. This framework should exploit background theory on conjunctive queries and query containment and minimization [1].

An architecture based on mediators is highly beneficial for deploying a SWIM. There exist two main approaches for integrating data sources [2] using mediator-based architectures: Global-as-View (GAV) [12] and Local-as-View (LAV) [9], [8]. The former provides descriptions of the global schema in terms of the views of local sources and relies on simple query reformulation techniques (i.e., query unfolding). The latter considers local sources as materialized views specified in terms of the global schema. LAV supports easily the evolution of the data integration system by just adding or removing the descriptions of local sources. In our work we advocate a hybrid approach called GLAV [6], which combines the previous advantages and exceeds the expressive power of both GAV and LAV.

2 A Motivating Example

Let's assume an XML source whose content is described by a DTD or an XML Schema (see Figure 1). XML data from this source contain information about Museums, exhibiting some artifacts for which we want to know their creator. Data stored in such sources can be queried using an XML query language like XPath [13], [14] or XQuery [15].

Now suppose that we add on top of this repository an RDF/S schema from the cultural domain. This mediated RDF/S schema can be queried using RQL and it can be used for defining personalized views with the help of RVL. However, since there are no actual RDF data, we need to reformulate the RQL queries expressed against this virtual RDF/S schema into queries appropriate for our XML source. For example, the following RQL query:

```
SELECT X
FROM {X}exhibits{Y}, {Y}denom{Z}
WHERE Z = "Louvre"
```

will be *reformulated* to the following XQuery query:

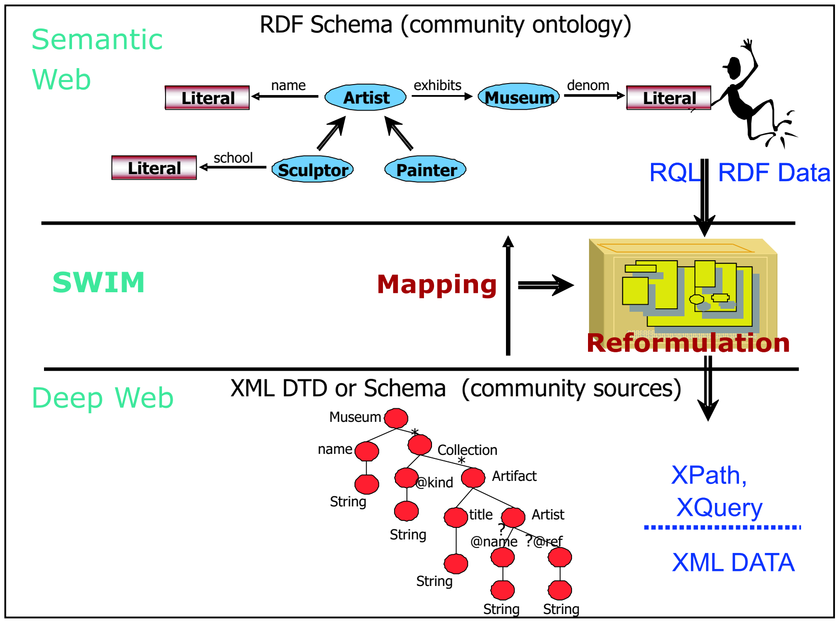


Fig. 1. Republishing XML as RDF

```
<RDF>
{
  <Bag>
  {
    for $var0 in document("art.xml")//Museum
    for $var1 in $var0/name
    for $var2 in $var0//Artist/@name
    where $var1/text()="Louvre"
    return <li>{$var2/text()}</li>
  }
  </Bag>
}
</RDF>
```

This reformulation involves several challenging issues. First of all, the schemas employed by our XML sources and the RDF/S mediator are different. Their discrepancies, usually called *heterogeneity conflicts*, can be classified under three axes: syntactic, structural and semantic [11]. As we can see in our example (Figure 1), we need to view the XML data through the RDF data model (syntactic conflict), to resolve categorization conflicts, given that in RDF/S there is a class hierarchy while in XML there isn't (structural conflict), as well as address naming mismatches; for instance the "name" of a Museum in XML is called "denomination" in RDF/S (semantic conflict).

In order to reconciliate the heterogeneous representations of our data we need to define appropriate *mapping rules*. Choosing an expressive but tractable logical framework to map data from XML to RDF/S is crucial for the success of a SWIM. These mappings are used for reformulating queries issued against the virtual RDF/S schema into queries acceptable by our XML sources. However, more complex mappings (in order to increase expressiveness) render query reformulation harder. Query reformulation becomes more complex if we take into consideration the presence of constraints capturing the semantics of both the RDF/S and XML data models, as well as, application-specific constraints coming from the schema (if any) of XML sources (like keys, foreign keys etc.). So, there is a need for a sound and complete reformulation algorithm.

Since reformulated queries are evaluated to remote sources and mediator queries resulting from automated manipulation/generation may entail redundancies, their optimization is crucial. In particular, optimization tries to eliminate redundant queries and to simplify queries by removing redundant predicates.

3 Contributions

In this context, we propose a middleware called ICS-FORTH SWIM (Semantic Web Integration Middleware) supporting the following functionalities:

3.1 A Formal Framework for Mapping Specification

As discussed previously, choosing a logical framework for defining the mappings is of great importance. Our idea was to represent both RDF and XML data models as first-order logic predicates and capture their semantics through appropriate constraints. In this way we reduce RDF to XML query reformulation problem to the relational equivalent one, and thus, we can reuse existing techniques for relational query containment and minimization.

More precisely, we rely on Linear Datalog (non-recursive Datalog without negation) for establishing the mappings and translating the RQL/RVL queries and views issued against the virtual RDF/S schema. The head of the Datalog rules consists of a conjunction of view clauses employed by the RDF/S view definition language RVL, and the body consists of XPath atoms that facilitate querying tree-structured XML sources. The former is used in order to point out the instantiation of RDF/S schemas with appropriate resources residing in our XML data sources. Employing some non interpreted built-in predicates (e.g., concat, split) for handling more intricate cases (like complex keys or string manipulation) enhances the expressiveness of the mappings.

As far as these mappings are concerned, they are interpreted in a constraint - oriented way implementing the GLAV approach of our middleware. We can map a *view* over the global RDF/S to a *view* over local XML sources, and each of these mappings is captured with the help of constraints. Constraints describe both the RDF/S schema in terms of the XML sources (GAV) and the XML sources in terms of the global RDF/S schema (LAV).

3.2 Query Reformulation and Optimization

Another powerful functionality of the SWIM is its ability to reformulate queries. RQL queries, expressed in terms of the RDF/S virtual schema, result in minimized queries expressed in terms of the XML sources by gradually applying a number of chasing/backchasing [3], [4], [5] steps. The use of this algorithm is proven to be sound and complete for disjunctions of conjunctive queries in the presence of disjunctive embedded dependencies (DEDs). This is guaranteed by the use of the Chase/Backchase algorithm given this type of input.

A query is chased with the help of the constraints that have been defined to express the semantics of the XML and RDF data models and in addition, with constraints (if any) coming from the XML data sources i.e., specification of keys and foreign keys, as well as, of domain constraints (e.g., enumerated types). The result of chasing is backchased for producing a minimal reformulation. These minimized queries are simplified (retaining as few predicates as possible) and the redundant ones are eliminated. In this way we guarantee that we query the XML sources with the minimum possible queries. Finally, the minimized reformulated queries are translated into XPath and/or XQuery.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison - Wesley, 1995.
2. Leopoldo Bertossi and Loreto Bravo. *Inconsistency Tolerance in Knowledgebases, Databases and Software Specifications*, chapter Consistent Query Answers in Virtual Data Integration Systems. Springer, 2004.
3. Alin Deutsch and Val Tannen. Querying XML with Mixed and Redundant Storage. Technical report, University of Pennsylvania, 2002.
4. Alin Deutsch and Val Tannen. MARS: A System for Publishing XML from Mixed and Redundant Storage. In *Proceedings of the 29th VLDB Conference, Berlin, Germany*, 2003.
5. Alin Deutsch and Val Tannen. Reformulation of XML Queries and Constraints. In *Proceedings of the International Conference on Database Theory (ICDT)*, 2003.
6. Marc Friedman, Alon Levy, and Todd Millstein. Navigational Plans for Data Integration. In *Proceedings of the sixteenth national conference on artificial intelligence and eleventh innovation applications of AI conference on Artificial intelligence and innovative applications of artificial intelligence*, pages 67–73, 1999.
7. G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. RQL: A Declarative Query Language for RDF. In *Proceedings of the 11th International World Wide Web Conference (WWW)*, Honolulu, Hawaii, 2002.
8. A. Levy. Answering Queries Using Views: A Survey. *The International Journal on Very Large Data Bases*, 2001.
9. Alon Y. Levy. Logic-Based Techniques in Data Integration. In Jack Minker, editor, *Workshop on Logic-Based Artificial Intelligence, Washington, DC, June 14-16, 1999*, College Park, Maryland, 1999. Computer Science Department, University of Maryland.
10. Aimilia Magkanaraki, Val Tannen, Vassilis Christophides, and Dimitris Plexousakis. Viewing the Semantic Web Through RVL Lenses. In *Proceedings of*

the Second International Semantic Web Conference (ISWC'03), Sanibel Island, Florida, USA, 20-23 October, 2003.
11. Amit Sheth and Vipul Kashyap. So Far (Schematically) yet So Near (Semantically). In *Proceedings of the IFIP WG 2.6 Database Semantics Conference on Interoperable Database Systems (DS-5)*, 1992.
12. J. Ullman. Information Integration Using Logical Views. *Theoretical Computer Science*, 239(2):189–210, 2000.
13. XML Path Language (XPath) 1.0. http://www.w3.org/tr/xpath/.
14. XML Path Language (XPath) 2.0. http://www.w3.org/tr/xpath20/.
15. XQuery 1.0: An XML Query Language. http://www.w3.org/tr/xquery/.

Semantic Query Routing and Processing in P2P Digital Libraries

George Kokkinidis[1], Lefteris Sidirourgos[1], Theodore Dalamagas[2], and Vassilis Christophides[1]

[1] Institute of Computer Science - FORTH
Vassilika Vouton, PO Box 1385, GR 71110, Heraklion, Greece and
Department of Computer Science, University of Crete
GR 71409, Heraklion, Greece
{kokkinid, lsidir, christop}@ics.forth.gr

[2] School of Electr. and Comp. Engineering,
National Technical University of Athens, Greece
{dalamag}@dblab.ece.ntua.gr

Abstract. This paper investigates the peer-to-peer (P2P) resource-sharing paradigm for highly distributed Digital Libraries (DL). The objective is to support decentralized sharing of data and services in a network of autonomous and heterogeneous DL nodes. P2P DLs can operate without a central coordination and offer important advantages such as a very dynamic environment where peers can join and leave the network at any time, while the network can scale up to a large number of peers. The advanced structuring and retrieval functionality of peers poses new challenges in query routing and processing over autonomous, distributed and dynamic networks of DL. The paper considers two fundamental aspects of P2P Digital Libraries (P2P DLs): query routing and processing. Specifically, we design and implement effective and efficient query routing in P2P DLs, exploiting intensional indexing of DL node views. Also, we study interleaved query routing and processing algorithms in P2P DLs to produce as quickly as possible the first query results.

1 Introduction

The digital library community envisions the availability of digital content on a global scale through Digital Libraries (DL) that can be accessed, integrated and individualized for any user, anytime and anywhere. A key point in such a vision is the interaction with multiple DL nodes to support integrated access. We believe that such interaction is far beyond the traditional information integration technologies, which impose restrictions on representation and communication languages used at both the semantic and the structural levels, since:

1. DL nodes should be autonomous. Ideally, a node must not have restrictions on how to organize its data and what kind of query capabilities to offer.

2. DL services should support decentralized sharing and management of data through a network of DL nodes. In such a network, a DL node must be able to provide data to other DL nodes and, at the same time, to have access to data of other DL nodes.
3. The diversity of DL nodes in terms of availability, processing power and interface options, makes a DL network a highly heterogeneous environment in terms of hardware/software setup in addition to the data being provided.
4. Finally, the system needs to be evolving in the sense of DL nodes joining and leaving the network at their own will. DL node arrivals and departures affect the data that is available.

Our work explores the application of the peer-to-peer (P2P) paradigm in DL technologies. In particular, schema-based P2P systems [2, 6] exploit schema information to specify what kind of data is provided by the involved peers. The advantages of this approach lies to the fact that (a) more sophisticated than keyword-based queries can be posed and (b) more efficient approaches can be developed for identifying peers that are capable of answering the queries. A natural candidate for representing descriptive schemas of information resources (ranging from simple structured vocabularies to complex reference models [8]) is the Resource Description Framework/Schema Language (RDF/S). RDF schemas offer rich semantics. The primitives of RDF schemas are classes and properties. Classes describe general concepts or entities. Properties describe the characteristics of classes or the relationships between classes.

RDF/S (a) enables a *modular design* of descriptive schemas based on the mechanism of *namespaces*; (b) allows easy *reuse* or *refinement* of existing schemas through *subsumption* of both class and property definitions; (c) supports partial descriptions since *properties* associated with a resource are by default *optional and repeated* and (d) permits *super-imposed descriptions* in the sense that a resource may be multiply classified under several classes from one or several schemas. These modelling primitives are crucial for schema-based P2P systems where monolithic RDF/S schemas and resource descriptions cannot be constructed in advance and DL nodes may have only incomplete descriptions about the available resources.

The advanced structuring and retrieval functionality of schema-based P2P systems raises new challenges for view integration, query routing and processing over autonomous, distributed and dynamic networks of DLs.

The main contributions of our work presented in this paper are (a) the design and implementation of effective and efficient query routing in P2P DLs, exploiting intensional indexing of DL node views and (b) the study of interleaved query routing and processing algorithms in P2P DLs in order to produce as quickly as possible the first query results.

1.1 Related Work

Several projects address query processing issues in general P2P systems [9, 7]. However, they require a priori knowledge of the relevant to a query peers. Mutant Query Plans (MQPs) [10] implement efficient query routing. Unlike our approach, MQP reduces the optimization opportunities by simply migrating possibly big XML fragments of query plans along with partial results of subqueries. In [11] indices are used to identify peers that can handle containment queries (e.g., in XML). However there are no details on how a set of semantically related peers can actually execute a complex query involving vertical and horizontal distribution. RDFPeers [12] is a scalable distributed RDF/S repository which efficiently answers multi-attribute and range queries. This approach ignores RDF/S schema information during query routing, while distributed query processing and execution policies are not addressed. In [13], a P2P architecture is introduced, based on the extension of an existing RDF/S store. Although schema information is used for indexing, RDF/S class and property subsumption is not considered. A schema-based P2P infrastructure for the Semantic Web is described in [6]. Their approach involves exact matching of basic class and property pattern and does not consider run-time adaptability of query plans.

2 P2P Digital Libraries

In order to design an efficient P2P DL infrastucture we need to address the following issues: (a) How DL nodes advertise their bases?, (b) How DL nodes formulate queries?, (c) How DL nodes route queries?, and (d) How DL nodes process queries?

2.1 Advertisements of DL Nodes

A schema-based P2P DL infrastructure requires that each DL node advertises its local base content to other DL nodes. Using these advertisements, a DL node becomes aware of the bases hosted by other nodes in the DL. In our approach, we assume that there are global RDF/s schemas for various communities, in which DL nodes have access through the mechanism of namespaces. However, a global RDF/S schema may contain numerous classes and properties not necessarily populated in a DL node. Therefore, we need a fine-grained definition of schema-based advertisements. We employ *RVL views* [5] to specify the subset of a community RDF/S schema(s) for which all classes and properties are populated in a DL node base. These views may be broadcasted to (or requested by) other DL nodes, thus informing the rest of the P2P DL of the information actually available in the DL nodes.

2.2 Query Formulation in DL Nodes

In this work, queries and views in a P2P DL are formulated by nodes in the RQL/RVL [4, 5] language. RQL is a typed functional language in the form of OQL. It uniformly queries both RDF data descriptions and schemas. RVL extends RQL by supporting views on RDF/S. In RQL/RVL, class and property path patterns allow users to navigate through the RDF/S schema of a DL node to retrieve resources. RQL queries allow us to retrieve the contents of any DL node base, namely resources classified under schema classes or associated to other resources using schema properties. It is worth noticing that RQL queries imply both intensional (i.e., schema) and extensional (i.e., data) filtering conditions.

2.3 Query Routing in P2P Digital Libraries

Query routing is responsible for finding the relevant to a query DL nodes (or more precisely their views) by taking into account data distribution (vertical, horizontal and mixed) of their bases committing to an RDF schema. The query-routing algorithm takes as input a query and the available DL node views and detects which DL nodes can actually answer the query as a whole or fragments of it. The latter is important, since there might be answers that can be received by joining partial answers from different DL nodes. Our approach exploits query/view subsumption algorithms [1] to check whether the classes or properties of the view are subsumed by the respective classes or properties used in the query. In this way, query routing takes into consideration semantic information from the RDF Schemas of the involved DL nodes.

Specifically, a fragmentor breaks the given query into subqueries, whose number is bounded by an input variable. The query/view subsumption algorithms of [1] are employed to determine which part of a query can be answered by a DL node view. For maintaining a distributed catalog of views published by the DL nodes in a P2P DL, appropriate DHT structures have been designed.

2.4 Query Processing in P2P Digital Libraries

Query processing is responsible for generating query plans according to the results returned by the routing algorithm (i.e., which DL nodes can actually answer the query as a whole or fragments of it). If more than one DL nodes can answer the same query fragment, the results from each such DL node base are "unioned" (horizontal distribution). The results obtained for different query fragments that are connected at a specific domain or range class are "joined" (vertical distribution). The generated query plan reflects the data distribution of the system and uses it for obtaining at execution time both complete and correct results.

The resulting query plan can be optimized. Compile-time optimization relies on algebraic equivalences (e.g., distribution of joins and unions) and heuristics allowing us to push, as much as, possible query evaluation to the same DL nodes. Additionally, cost-based optimizations based on statistics about the DL node bases enable to reorder joins and choose between different execution policies for the query plans (e.g., data versus query shipping).

A key feature of our approach is that *query routing and processing are interleaved* in several iteration steps. This leads to the creation and execution of multiple query (sub)plans that when "unioned" offer completeness in the results. Specifically, starting with the initial query, at each iteration step, smaller subqueries are considered in order to find the relevant DL nodes (i.e., routing) that can actually answer them (i.e., processing). The routing information, i.e., remote DL node views, is acquired by the lookup service offered by the system on top of intensional DHT structures. The interleaved query evaluation terminates when the initial query is decomposed into its basic class and property patterns.

The main advantage of the interleaved query routing and processing algorithm is that the query results are collected as quickly as possible since they require fewer intra-DL node joins. More precisely, each query fragment is looked up as a whole and only DL nodes that can fully answer it are actually involved in each query processing iteration step.

References

1. Christophides V, Karvounarakis G, Koffina I, Kokkinidis G, Magkanaraki A, Plexousakis D, Serfiotis G, Tannen V (2003) The ICS-FORTH SWIM: A Powerful Semantic Web Integration Middleware. In Proc. of the 1st Int'nal Workshop on Semantic Web and Databases (SWDB), Berlin, Germany.
2. Halevy AY, Ives ZG, Mork P, Tatarinov I (2003) Piazza: Data Management Infrastructure for Semantic Web Applications. In Proc. of the 12th Int'nal World Wide Web Conf. (WWW).
3. Ives ZG (2002) Efficient Query Processing for Data Integration. phD Thesis, University of Washington.
4. Karvounarakis G, Alexaki S, Christophides V, Plexousakis D, Scholl M (2002) RQL: A Declarative Query Language for RDF. In Proc. of the 11th Int'nal World Wide Web Conf. (WWW), Honolulu, Hawaii, USA.
5. Magkanaraki A, Tannen V, Christophides V, Plexousakis D (2003) Viewing the Semantic Web Through RVL Lenses. In Proc. of the 2nd Int'nal Semantic Web Conf. (ISWC).
6. Nejdl W, Wolpers M, Siberski W, Schmitz C, Schlosser M, Brunkhorst I, Loser A (2003) Super-Peer-Based Routing and Clustering Strategies for RDF-Based P2P Networks. In Proc. of the 12th Int'nal World Wide Web Conf. (WWW), Hungary.
7. Sahuguet A (2002) ubQL: A Distributed Query Language to Program Distributed Query Systems. phD Thesis, University of Pennsylvania.
8. Magkanaraki A, Alexaki S, Christophides V, Plexousakis D (2002) Benchmarking RDF Schemas for the Semantic Web. In Proc. of the 1st Int'nal Semantic Web Conf. (ISWC'02).
9. Kemper A, Wiesner C (2001) HyperQueries: Dynamic Distributed Query Processing on the Internet. In Proc. of the Int'nal Conf. on Very Large Data Bases (VLDB), Rome, Italy.
10. Papadimos V, Maier D, Tufte K (2003) Distributed Query Processing and Catalogs for P2P Systems. In Proc. of the 2003 CIDR Conf.
11. Galanis L, Wang Y, Jeffery SR, DeWitt DJ (2003) Processing Queries in a Large P2P System. In Proc. of the 15th Int'nal Conf. on Advanced Information Systems Engineering (CAiSE).
12. Cai M, Frank M (2004) RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network. In Proc. of the 13th Int'nal World Wide Web Conf. (WWW), New York.
13. Stuckenschmidt H, Vdovjak R, Houben G, Broekstra J (2004) Index Structures and Algorithms for Querying Distributed RDF Repositories. In Proc. of the Int'nal World Wide Web Conf. (WWW), New York, USA.

Publish/Subscribe Functionalities for Future Digital Libraries using Structured Overlay Networks

Christos Tryfonopoulos   Stratos Idreos   Manolis Koubarakis

Dept. of Electronic & Computer Engineering
Technical University of Crete, 73100 Chania, Crete, Greece
{trifon,sidraios,manolis}@intelligence.tuc.gr

1 Introduction

We are interested in the problem of *distributed resource sharing* in future digital libraries (DLs). We adopt a pure P2P architecture (illustrated in Figure 1), but our ideas can be easily modified to work in the case of hierarchical P2P networks, as in [3]. *Information providers* (DLs) and *information consumers* (users) are both represented by peers participating in a peer-to-peer (P2P) overlay network. There are two kinds of basic functionality that we expect this architecture to offer: *information retrieval (IR)* and *publish/subscribe (pub/sub).* In an IR scenario a user poses a *query* (e.g., "I am interested in papers on bio-informatics") and the system returns information about matching resources. In a pub/sub scenario (also known as *information filtering (IF)* or *selective dissemination of information (SDI))* a user posts a *subscription* (or *profile* or *continuous query*) to the system to receive notifications whenever certain events of interest take place (e.g., when a paper on bio-informatics becomes available).

In this extended abstract we concentrate on the latter kind of functionality (pub/sub) and sketch how to provide it by extending the distributed hash table Chord [4]. *Distributed Hash Tables (DHTs)* are the second generation *structured* P2P overlay networks devised as a remedy for the known limitations of earlier P2P networks such as Napster and Gnutella. We present a set of protocols, collectively called *DHTrie*, that extend the Chord protocols with pub/sub functionality.

We assume that resources are annotated using a well-understood attribute-value model called $\mathcal{AWPS}$ in [2]. Thus publications and subscriptions will also be expressed in $\mathcal{AWPS}$. $\mathcal{AWPS}$ is based on *named attributes* with value *free text* interpreted under the Boolean and vector space (VSM) models. The query language of $\mathcal{AWPS}$ allows Boolean combinations of comparisons A *op* v, where A is an attribute, v is a text value and *op* is one of the operators "equals", "contains" or "similar" ("equals" and "contains" are Boolean operators and "similar" is interpreted using the VSM or LSI model). The following is an example of a publication in $\mathcal{AWPS}$:

$$\{\ (AUTHOR, \text{“}\textit{John Smith}\text{”}),\ (TITLE, \text{“}\textit{Information dissemination in P2P ...}\text{”}),\\ (ABSTRACT, \text{“}\textit{In this paper we show that ...}\text{”})\ \}$$

The following is an example of a query:

$$(AUTHOR = \text{“}\textit{John Smith}\text{”})\ \wedge\ (TITLE \sqsupseteq P2P \wedge (information \prec_{[0,0]} alert))\ \wedge\\ (ABSTRACT \sim_{0.7} \text{“}\textit{P2P architectures have been...}\text{”})$$

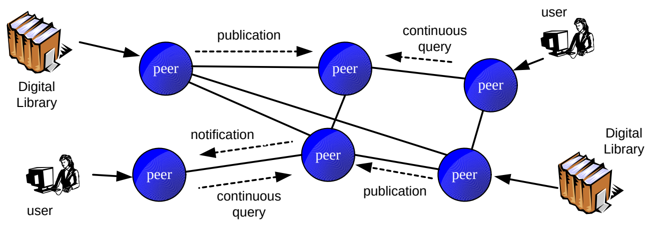


Fig. 1. Distributed resource sharing in future DLs

This query requests resources that have *John Smith* as their author, and their title contains the word *P2P* and a word pattern where the word *information* is immediately followed by the word *alert*. Additionally, the resources should have an abstract similar to the text value *"P2P architectures have been ..."* with similarity greater than 0.7. The contributions of this abstract are the following.

The research presented in this abstract is a continuation of our previous work on the DL information alert architecture DIAS [2] and the system P2P-DIET [1]. The main contribution of our current work is that our protocols are extensions of DHTs and achieve much better scalability, robustness, fault-tolerance and load balancing.

The rest of this extended abstract is as follows. Section 2 gives some details of the DHTrie protocols and Section 3 discusses very briefly our experimental evaluation of DHTrie.

2 The DHTrie Protocols

We implement pub/sub functionality by a set of protocols called the *DHTrie protocols* (from the words DHT and trie). The DHTrie protocols use *two levels of indexing* to store queries submitted by clients. The first level corresponds to the partitioning of the global query index to different nodes using DHTs as the underlying infrastructure. Each node is responsible for a fraction of the submitted user queries through a mapping of attribute values to node identifiers. The DHT infrastructure is used to define the mapping scheme and also manages the routing of messages between different nodes. We use an extension of the Chord DHT [4] to implement our network. The set of protocols that regulate node interactions are described in the next sections.

The second level of our indexing mechanism is managed locally by each node and is used for indexing the user queries the node is responsible for. In this level, each node uses a hash table to index all the atomic queries contained in a query by using their attribute name as the key. For each atomic Boolean query the hash table points to a *trie*-like structure that exploits *common words* and a hash table that indexes text values in equalities as in [6]. Additionally for atomic VSM queries an inverted index for the most "significant" query words is used as in [7].

In this abstract we will focus on the first level of indexing and present the subscription, publication and notification protocols that regulate node interactions. Protocols for query updating and removal are omitted due to space. The local indexing algorithms we use and their experimental evaluation are thoroughly discussed in [6, 7].

2.1 The Subscription Protocol

Let us assume that a node P wants to submit a query q of the form:

$$\bigwedge_{i=1}^{m} A_i = s_i \ \wedge \ \bigwedge_{i=m+1}^{n} A_i \sqsupseteq wp_i \ \wedge \ \bigwedge_{i=n+1}^{k} A_i \sim_{a_i} s_i$$

To do so, P randomly selects a single word w contained in any of the text values $s_1, \ldots, s_m, s_{n+1}, \ldots, s_k$ or word patterns $wp_{m+1}, \ldots, wp_n$ and computes $H(w)$ to obtain the identifier of the node that will be responsible for query q. Then P creates message $\textsc{FwdQuery}(id(P), IP(P), qid(q), q)$, where $qid(q)$ is a unique query identifier assigned to q by P and $IP(P)$ is the IP address of P. This message is then forwarded in $O(logN)$ steps to the node with identifier $H(w)$ using the routing infrastructure of the DHT. Notice that $id(P)$ and $IP(P)$ need to be sent to the node that will store Q to facilitate notification delivery (see Section 2.3).

When a node P' receives a message FwdQuery containing q, it stores q using the second level of our indexing mechanism. P' uses a hash table to index all the atomic queries of q, using as key the attributes $A_1, \ldots, A_k$. To index each atomic query, three different data structures are also used: (i) a hash table for text values $s_1, \ldots, s_m$, (ii) a trie-like structure that exploits common words in word patterns $wp_{m+1}, \ldots, wp_n$, and (iii) an inverted index for the most "significant" words in text values $s_{n+1}, \ldots, s_k$. P' utilises these data structures at filtering time to find quickly all queries q that match an incoming publication p. This is done using an algorithm that combines algorithms BestFitTrie [6] and SQI [7].

2.2 The Publication Protocol

When a node P wants to publish a resource, it first constructs a publication $p = \{(A_1, s_1), (A_2, s_2), \ldots, (A_n, s_n)\}$ (the resource description). Let $D_1, \ldots, D_n$ be the sets of *distinct* words in $s_1, \ldots, s_n$. Then publication p is sent to *all* nodes with identifiers in the list $L = \{H(w_j) : \ w_j \in D_1 \cup \cdots \cup D_n\}$. The subscription protocol guarantees that L is a superset of the set of identifiers responsible for queries that match p.

The propagation of publication p in the DHT proceeds as follows. P removes duplicates from L and sorts it in ascending order clockwise starting from $id(P)$. This way we obtain less identifiers than the distinct words in $D_1 \cup \cdots \cup D_n$, since a node may be responsible for more than one words contained in the document. Having obtained L, P creates a message $\textsc{FwdResource}(id(P), pid(p), p, L)$, where $pid(p)$ is a unique metadata identifier assigned to p by P, and sends it to node with identifier equal to $head(L)$ (the first element of L). This forwarding is done by the following *recursive* method: message FwdResource is sent to a node P', where $id(P')$ is the greatest identifier contained in the finger table of P, for which $id(P') \leq head(L)$ holds.

Upon reception of a message FwdResource by a node P, $head(L)$ is checked. If $id(P) < head(L)$ then P just forwards the message as described in the previous paragraph. If $id(P) \geq head(L)$ then P makes a copy of the message, since this means that P is one of the intended recipients contained in list L (in other words P is responsible for key $head(L)$). Subsequently the publication part of this message is matched with the node's local query database using the algorithm mentioned in Section 2.1 and the appropriate subscribers are notified (see Section 2.3). Additionally list L is modified to L' in the following way. P deletes all elements of L that are smaller than $id(P)$ starting from $head(L)$, since all these elements have P as their intended recipient. In the new list L' that results from these deletions we have that $id(P) < head(L')$. This happens because in the general case L may contain more than one node identifiers that are managed by P (these identifiers are all located in ascending order at the beginning of L). Finally, P forwards the message to node with identifier $head(L')$.

2.3 The Notification Protocol

When a message FwdResource containing a publication p of a resource arrives at a node P, the queries matching p are found by utilising its local index structures and using the algorithms briefly described in Section 2.1.

Once all the matching queries have been retrieved from the database, P creates notification messages of the form QNotification$(l(r))$ and contacts all the nodes that their queries where matched against p using their IP address associated with the query they submitted. If a node P' is not online when P tries to notify it about the published resource, the notification message is sent to the $successor(P')$. In this way P' will be notified the next time it logs on the network. The modifications to the join and leave protocols of Chord to achieve this functionality originally presented in the non-DHT system P2P-DIET [1] are omitted due to space considerations. To utilise the network in a more efficient way, notifications can also be batched and sent to the subscribers when traffic is expected to be low.

2.4 Frequency Cache

In this section we introduce an additional routing table that is maintained in each node. This table, called *frequency cache (FCache)*, is used to reduce the cost of publishing a resource by storing the IP addresses of the nodes responsible for frequent words contained in published documents. FCache is a hash table used to associate each word that appears in a published document with a node IP address. FCache uses a word w as a key, and each FCache entry is a data structure that holds an IP address. Thus, whenever P needs to contact another node P' that is responsible for queries containing w, it searches its FCache. If FCache contains an entry for w, P can directly contact P' using the IP stored in its FCache. If w is not contained in FCache, P uses the standard DHT lookup protocol to locate P' and stores contact information in FCache for further reference. Using FCache the cost of processing a published resource p is reduced to $O(v + (h - v) \log N)$, where v is the number of words of p contained in FCache.

FCache entries are populated as follows. Each time a resource p is published at a node P, P contacts the nodes responsible for storing queries with words contained in p, as we described in Section 2.2. After this process is over, P knows the contact information (namely the IP address) of those nodes, and stores it to FCache along with the word each node is responsible for. After that, for each publication taking place at P, P maintains this routing information for the most frequent words contained in resources published to it. Notice that the construction and maintenance of FCache is based only on local information and that the only extra cost involved is FCache misses (which cost $O(logN)$ and the routing information discovered is also cached for further reference).

3 Brief Presentation of Experimental Results

We have evaluated DHTrie experimentally in a distributed digital library scenario with hundreds of thousands of nodes and millions of user profiles. For our experiments we used 10426 documents downloaded from CiteSeer and also used in [6]. The documents are research papers in the area of Neural Networks and we will refer to them as the NN corpus. Because no database of queries was available to us, our queries are synthetically generated by exploiting 2000 documents of the corpus. The remaining 8426 documents are used to generate publications.

Our experiments show that the DHTrie protocols are *scalable*: the number of messages it takes to publish a document and notify interested subscribers remains almost constant as the network grows. Moreover, the increase in message traffic shows little sensitivity to increase in document size. We demonstrate that simple data structures with only local information can make a big difference in a DHT environment: the routing table FCache manages to reduce network traffic by a factor of 4 in all the alternative methods we have studied.

Since probability distributions associated with publication and query elements are expected to be skewed in typical pub/sub scenarios, achieving a *balanced load* is an important problem. We have studied an important case of load balancing for DHTrie and present a new algorithm which is also applicable to the standard DHT look-up problem.

The details of our experiments appear in [5] and will be discussed in detail in our workshop presentation.

References

1. S. Idreos, M. Koubarakis, and C. Tryfonopoulos. P2P-DIET: An Extensible P2P Service that Unifies Ad-hoc and Continuous Querying in Super-Peer Networks. In *Proc. of SIGMOD*, 2004. Demo paper.
2. M. Koubarakis, T. Koutris, P. Raftopoulou, and C. Tryfonopoulos. Information Alert in Distributed Digital Libraries: The Models, Languages and Architecture of DIAS. In *Proc. of ECDL*, 2002.
3. J. Lu and J. Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *Proc. of ECIR*, 2005. To appear.
4. I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of ACM SIGCOMM*, 2001.
5. C. Tryfonopoulos, S. Idreos, and M. Koubarakis. Publish/Subscribe Functionality in IR Environments using Structured Overlay Networks. Submitted to a conference.
6. C. Tryfonopoulos, M. Koubarakis, and Y. Drougas. Filtering Algorithms for Information Retrieval Models with Named Attributes and Proximity Operators. In *Proc. of ACM SIGIR*, 2004.
7. T.W. Yan and H. Garcia-Molina. The SIFT information dissemination system. *ACM TODS*, 24(4):529–565, 1999.

Information Access in Digital Libraries: Steps Towards a Conceptual Schema

Carlo Meghini[1] and Nicolas Spyratos[2]

[1] CNR – ISTI, Pisa, Italy, meghini@isti.cnr.it
[2] Universitè de Paris Sud – LRI, Paris, France, spyratos@lri.fr

Abstract. Two basic factors that influence the quality of information access are the *model* of system/user interaction and the *mode* of interaction. The model establishes the language of communication, i.e., the basic terms in which the user interacts with the system, while the mode establishes the role played by each actor (passive or active) during interaction. More precisely, the model provides support for representing the objects to be accessed and the information needs of the user, as well as for the mapping of a (possibly ordered) set of objects to each information need. (I suppose that what you mean here is: information need = query and mapping=query answering) As for the interaction mode, it depends on which of the two actors is passive and which is active. In query-answering systems, only the user is active; in filtering systems, only the system is active (in the sense that it selects from a stream of incoming objects, only those that are relevant to the user); in personalized query-answering systems and in recommendation systems, both the user and the system are active. In this paper, we attempt a classification of information access techniques for digital libraries, based on the two aforementioned parameters, namely the model and the mode used for system/user interaction.

1 Introduction

A digital library (DL) system comprises a large, distributed information space, where the objects carrying the knowledge required by the users, live. Setting up and maintaining a DL would be entirely useless if users were not provided with the adequate tools to access the DL information contents.

In defining the information access service of a DL, there are at least 3 areas of the information system field which can give significant contributions:

– *information retrieval* (IR) can contribute content-based methods, that is methods that exploit the signal (or form, or syntax) level of an information object [2]. For textual objects, these include all classical methods based on the statistical properties of language, such as the vector-space method and the probabilistic method. For non-textual objects, in the last decade there has been a fluorishing of methods for similarity-based retrieval of images, video [4] and audio objects.

– The *database* area can contribute all methods for attribute-based information querying, ranging from traditional databases, to semi-structured databases; the methods falling under the last category are especially useful when addressing the *structure* of objects.
– *Knowledge representation* can contribute methods for accessing objects by querying descriptions of their contents (in the sense of meaning); these descriptions are typically couched in terms of representations of the underlying domain of discourse, or *ontologies* as these have come to be termed lately.

Putting the techniques found in the above fields all together at work, requires a conceptual schema which integrates in a unique framework the different aspects of DL objects addressed by each of them. In what follows we will try to sketch such a schema.

2 A conceptual schema of information access

We characterize the space of information access by the following, orthogonal dimensions:

1. the information access model, which establishes the basic terms of the system/user interaction;
2. the information access mode, which establishes the role of the user and that of the system during their interaction.

2.1 Information access models

Similarly to an information retrieval model, an information access model specifies: (1) a representation of the objects to be accessed; (2) a representation of the user information needs; and (3) a function associating a (possibly ordered) set of objects to each user information need. Information access models can be categorized according to the options available on each one of their dimensions.

For object representation, following [6] we distinguish between simple objects and composite objects. Simple objects cannot be further decomposed, and can be represented along the following dimensions:

– *content abstractions*: these are representations used to access objects via IR techniques. They are created by IR indexers in an automatic way, by extracting low-level features from objects, such as the number of word occurrences in a text, or the energy level in a certain region of an image); content abstractions retain that part of the information originally present in the object content that is considered sufficient to characterize the object for access purposes;
– *content representations*: these are symbolic representations of the meaning of documents, that is descriptions formulated in some suitable knowledge representation language, spelling out the truth conditions of the object. Content representations can be constructed manually, sometimes with the assistance

of some knowledge extraction tool, or automatically, for instance via object classification methods. They can be precise or uncertain, depending on whether or not the content representation formulas are expressed in an uncertain logic, such as fuzzy or probabilistic logic.

Composite objects are structured set of simple objects, thus they are typically represented as mathematical *structures*, reflecting their internal organization.

Finally, objects, whether simple or composite, can have a *profile*, that is an attribute-based description of their external properties, such as the author and the publisher of a book, or the data and time at which an image has been taken, and the like.

In the context of digital libraries, the above 4 dimensions of object representation reduce to 3 only, since content representations and profiles are typically grouped together under the label of discovery metadata (simply metadata, hereafter). The metadata associated to a DL object can therefore vary from simple records (set of attribute-value pairs) typically adhering to some standardized schema (such as Dublin Core [1]), to very complex representations expressed in some knowledge representation language (such as OWL [5]) which must be coupled with a representation of the underlying domain of discourse (ontology), in order to be properly used for, e.g. information access. In summary, we will therefore consider object representation as being categorized as:

1. Content,
2. Metadata, and
3. Structure.

For information need representation, we have 2 options: formal language queries or natural language queries.

For the retrieval function, we have 2 options: exact match or best match.

Not all combinations of these options give rise to meaningful information access models. Considering the object representation options we have the following:

1. Content is typically addressed via IR techniques, based on natural language queries and best-match retrieval function. For instance, users access the content of an image base by providing as a query an image itself[3] In so doing, users are implicitly asking the system to retrieve images which "looks" similar to the image query, at least as far as the system can tell. For this reason, methods providing access by content are often called "*similarity-based*" access methods. An analogous pattern is found in access models for textual or for audio content.
   Notice that in some models of this kind, the query may not be expressed in the same medium as the sought objects. For instance, when accessing by content a video database, users may provide an image as a query, and the system is expected to retrieve the scenes sharing visual similarity with the provided image.

[3] We consider images, as well as spoken language, as expressions of a natural language.

Similarity-based access models are medium-dependent, in that different media require different techniques, and the possibility of using the same or a similar technique across different media is very limited. In addition, experience has shown that even within the same medium, the effectiveness of techniques is application-dependent: for example, retrieving by similarity sport images requires different techniques from those necessary to establish the similarity of X-ray images.

2. Metadata is typically addressed via formal language queries, thus we are in the realm of databases or knowledge representation, depending from the degree of sophisticatedness of the involved representations. For instance, to query a Dublin Core scheme, one needs a very simple language allowing to state basic relational conditions on simple numeric- or string-based attribute values; instead, to query an OWL representation the ability to state conditions involving taxonomies and navigation of graph structures is required.

   In general, a query to metadata takes the form of a logical formula, and an exact match retrieval function is employed for query evaluation, grounded in some logical theory. We call these models *semantic* access models.

   Some retrieval engines allow to exploit information (namely, text) retrieval techniques when accessing objects via their metadata records. This is achieved by seeing metadata records as pieces of text whose words are the attribute values. Such a text is treated in the same way as textual content, and matched against a natural language query as in a similarity-based textual access method. This kind of access model is widely employed on bibliographic records, as it frees users from the necessity of knowing the meaning of metadata attributes.
3. Also object structure can be queried in two different ways:
   - Via exact match retrieval functions, such as the one underlying the query language XPath [3]. In this case, the structural query is mostly embedded into a larger query addressing either content or metadata. We do not have therefore separate classes of access models but sophistications of similarity-based or semantic access models allowing also the specification of structural clauses in queries.
   - Via best match retrieval functions, which express in a quantitative way the degree of similarity between the structure of each object and a user provided structure class. These models have been investigated in the context of XML, and are a structural variation of similarity-based models. We call the models falling in this class "*similar structure-based* access models".

In summary, we have the following information access model categories:

1. similarity-based access models, possibly with structural conditions;
2. semantic access models, possibly with structural conditions; and
3. similar structure-based access models.

2.2 Information Access Modes

Independently from the access model, the information access modes establishes the role of the system and that of the user. Each of these two can be either passive or active. Excluding the case in which they are both passive, we have the following 3 types of information access systems:

1. (Non-personalized) query-answering systems: in this case the user is active, that is, poses a query, while the system is passive, that is it just evaluates the user query and returns the result.
2. Filtering systems, in which the user is passive, that is, does not pose any query, and the system is active, that is it selects on a stream of incoming objects, those that are deemed as relevant for the user; the relevance assessment is performed by relying on a user profile.
3. Personalized query-answering systems, in which the user is active, that is, poses a query, and the system is active too, in that it alters the query evaluation process by taking into account the user preferences, represented as a profile. Recommendation systems also fall into this category, since they propose recommendations along with the answer to the query.

3 Conclusions

On the basis of the analysis outlined in this paper, a categorization of information access in a DL can be derived. This categorization can be used as a basis for the development of an information access service offering a wide range of possibility for exploiting the contents of a DL.

References

1. Dublin core metadata element set. `http://dublincore.org/documents/dces/`, December 2004.
2. Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley Longman Publishing Co. Inc., 1999.
3. Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernndez, Michael Kay, Jonathan Robie, and Jérôme Simèon. XML path language (XPath) 2.0. `http://www.w3.org/TR/xpath20/`, October 2004.
4. Alberto Del Bimbo. *Visual Information Retrieval.* Morgan Kaufmann Publishers Inc., 1999.
5. Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language overview. `http://www.w3.org/TR/owl-features/`, February 2004. W3C Recommendation.
6. Carlo Meghini, Fabrizio Sebastiani, and Umberto Straccia. A model of multimedia information retrieval. *Journal of the ACM*, 48(5):909–970, 2001.

Challenges of Distributed Search Across Digital Libraries

Matthias Bender, Sebastian Michel, Gerhard Weikum, Christian Zimmer
{mbender, smichel, weikum, czimmer}@mpi-sb.mpg.de

Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany

Abstract. We present the MINERVA[1] project that tackles the problem of collaborative search across a large number of digital libraries. The search engine is layered on top of a Chord-style peer-to-peer overlay network that connects an a-priori unlimited number of peers or digital libraries. Each library posts a small amount of metadata to a conceptually global, but physically distributed directory. This directory is used to efficiently select promising libraries to execute a query based on their local data. The paper discusses current challenges regarding replication, caching and proactive dissemination, query routing based on local user profiles such as bookmarks, and benefit/cost models for query routing.

1 Introduction

The peer-to-peer (P2P) approach allows handling huge amounts of data of digital libraries in a distributed and self-organizing way. These characteristics offer enormous potential benefit for search capabilities powerful in terms of scalability, efficiency, and resilience to failures and dynamics. Additionally, such a search engine can potentially benefit from the intellectual input (e.g., bookmarks, query logs, click streams, etc.) of a large user community. However, recent research on structured P2P architectures is typically limited to exact-match queries on keys. This is insufficient for text queries that consist of a variable number of keywords, and it is absolutely inappropriate for full-fledged Web search where keyword queries should return a ranked result list of the most relevant approximate matches [7].

This paper builds upon the MINERVA system architecture presented in [4] and brings current challenges to ultimatively making distributed search across digital libraries feasible. MINERVA provides ranked search on data and and an efficient query mechanism that adheres to reasonable space and bandwidth limits. Unlike the approach criticized in [10], we do not spread inverted lists across the directory, but use only pointers to promising digital libraries as compact metadata and utilize these pointers to efficiently answer multiple-keyword queries. We leverage the extensive local indexes (which would be impossible to efficiently share across all peers) to incorporate features that are impossible in the approach studied in [10], such as phrase matching or proximity searches. Our

[1] Minerva is the Roman goddess of science, wisdom, and learning, and also happens to be a Greek underwear manufacturer.

bandwidth requirements are well within the postulation that a query should send no more data than the size of the documents ultimatively retrieved.

2 Related Work

Recent research on structured P2P systems, such as Chord [17], CAN [13], Pastry [15], or P-Grid [1] is typically based on various forms of distributed hash tables (DHTs) and supports mappings from keys to locations in a decentralized manner such that routing scales well with the number of peers in the system. In the following we briefly discuss some prior and ongoing projects towards P2P Web search.

Galanx [19] is a P2P search engine implemented using the Apache HTTP server and BerkeleyDB. The Web site servers are the peers of this architecture; pages are stored only where they originate from. PlanetP [8] is a publish-subscribe service for P2P communities, supporting content ranking search. PlanetP distinguishes local indexes and a global index to describe all peers and their shared information. The global index is replicated using a gossiping algorithm. The system appears to be limited to a few thousand peers.

Odissea [18] assumes a two-layered search engine architecture with a global index structure distributed over the nodes in the system. A single node holds the complete, Web-scale, index for a given text term (i.e., keyword or word stem). Query execution uses a distributed version of Fagin's threshold algorithm [9]. The system appears to cause high network traffic when posting document metadata into the network, and the presented query execution method seems limited to queries with at most two keywords. The paper actually advocates using a limited number of nodes, in the spirit of a server farm. The system outlined in [14] uses a fully distributed inverted text index, in which every participant is responsible for a specific subset of terms and manages the respective index structures. Particular emphasis is put on minimizing the bandwidth used during multi-keyword searches. [11] considers content-based retrieval in hybrid P2P networks. The peer selection for forwarding queries is based on the Kullback-Leibler divergence between peer-specific statistical models of term distributions.

In addition to this recent work on P2P search, prior research on distributed IR and metasearch engines is also potentially relevant; see [6, 20] for overviews. However, their work has assumed a relatively small number of digital libraries or databases and a fairly static configuration.

3 System Design

As a detailed description of the system design has already been given in [4], we only present a brief overview here and refer the interested reader to this prior work.

We view every library as autonomous. A conceptually global but physically distributed directory, which is layered on top of a Chord-style dynamic hash table (DHT) [17], holds only very compact, aggregated information about the peers' local indexes and only to the extent that the individual peers are willing to disclose. Every peer is responsible for a randomized subset of the global directory.

The global directory consists of aggregated information (*Posts*) that contains contact information about the digital library who posted this summary together with statistics to calculate IR-style relevance measures that try to estimate the relevance of a particular digital library to a query. These measures are used to support the *peer selection* process, i.e., determining the most promising libraries for a particular query.

If, at query time, the local query result is considered unsatisfactory by the user, a library retrieves a list of potentially useful libraries. Using collection selection methods from distributed IR and metasearch, a number of promising libraries for the complete query is computed from these PeerLists. In [3] we have studied promising and efficient techniques for this purpose. Subsequently, the query is forwarded to these libraries and executed based on their local indexes. The results from the various libraries are combined at the querying peer into a single result list; this step is referred to as *result merging*.

The goal of finding high-quality search results with respect to precision and recall cannot be easily reconciled with the design goal of unlimited scalability, as the best information retrieval techniques for query execution rely on large amounts of document metadata. Posting only compact, aggregated information about local indexes and using appropriate peer selection methods to limit the number of peers involved in a query keeps the global directory manageable and reduces network traffic. We expect this approach to scale very well as more and more peers jointly maintain the moderately growing global directory.

4 Challenges

Our work is driven by the question of how a collaborative search across digital libraries can benefit from the unique nature of the P2P paradigm. We want to address the shortcomings of centralized search engines and further benefit from the intellectual input of a large user community.

Democratic Community Search: To overcome the danger of the infiltration of a centralized index by (commercial) interest groups (as increasingly seen today on popular web search engines) we leverage the indexes from potentially thousands of digital libraries, making it harder to bias the final query results.

Implicit User Feedback: Additionally, we want to incorporate the fact that every library has its own local index, e.g., by executing all queries first locally at the initiating library and using implicit-feedback techniques for automated query expansion (e.g., using the well-known IR technique of pseudo relevance feedback [5] or other techniques based on query logs [12] and click streams [16]).

User Recommendations: We want to incorporate local user bookmarks into our query execution [2]. Bookmarks represent strong recommendations for specific documents. Also, user bookmarks can be considered as compact samples of peer indexes that describe their fields of interest. Queries could be exclusively forwarded to thematically related libraries with similarly interested users, to improve the chances of finding *subjectively* relevant pages.

Replication: In order to achieve service levels comparable to today's search engines and in order to actually benefit from the infrastructural advantage of a distributed system, we are going to introduce a certain degree of replication to the directory. Currently, as dictated by the DHT-style maintenance approach, a peer gracefully leaving the system forwards its share of the global directory to another peer. Analogously, a library entering the system asks for its appropriate share of the directory. While one approach to ensure a valid, complete, and up-to-date directory is to simply rely on the libraries to regularly re-send their Posts to the directory, it is also necessary to replicate parts of the directory to avoid data loss as libraries ungracefully leave the system. Also, replication can serve as a load-balancing measure, relieving the burden from peers that host popular parts of the directory, and increases data availability. Adaptive, self-tuning strategies for choosing appropriate degrees of replication and placing replicas are an open issue in our ongoing work.

Caching and Proactive Dissemination: To further enhance query efficiency, statistical summaries and also the results of queries from across the network may be cached in a way that allows other peers not only to instantly benefit from the existing query results but also to benefit from click streams that were recorded on the occasion of similar queries. Statistical summaries may also be disseminated proactively among thematically related peers. Finding good strategies to this end is a widely open issue.

Overlap-aware Query Routing: Peer selection has been a research issue for years. Most of the existing literature estimates the expected result quality of a collection, typically using precomputed statistics, and ranks the collections accordingly. We believe that this is insufficient if the collections overlap, e.g., in the scenario of digital libraries that share an arbitrarily large fraction of documents. We argue for the extension of existing quality measures using estimators of mutual overlap among collections. Preliminary experiments show that such a combination can outperform popular approaches based on quality estimation only, such as CORI [6]. Taking overlap into account during collection selection in this scenario can drastically decrease the number of libraries that have to be contacted in order to reach a satisfactory level of recall, which is a great step towards the feasibility of distributed search across digital libraries.

Benefit/Cost Optimization: Ultimatively, we want to introduce a sophisticated $benefit/cost$ ratio when selecting remote libraries for query forwarding. For the benefit estimation, it is intuitive to consider such measures as described in the previous paragraph. Defining a meaningful cost measure, however, is an even more challenging issue. While there are techniques for observing and inferring network bandwidth or other infrastructural information, expected response times (depending on the current system load) are changing over time. One approach is to create a distributed Quality-of-Service directory that, for all peers, holds moving averages of recent response times.

References

1. K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt. Improving data access in p2p systems. *IEEE Internet Computing*, 6(1):58–67, 2002.
2. M. Bender, S. Michel, G. Weikum, and C. Zimmer. Bookmark-driven query routing in peer-to-peer web search. In J. Callan, N. Fuhr, and W. Nejdl, editors, *Proceedings of the SIGIR Workshop on Peer-to-Peer Information Retrieval*, pages 46–57, 2004.

3. M. Bender, S. Michel, G. Weikum, and C. Zimmer. The minerva project: Database selection in the context of p2p search. *Accepted to BTW05; Available at http://www.mpi-sb.mpg.de/units/ag5/software/minerva/ publications/submitted05.pdf*, 2005.
4. M. Bender, S. Michel, C. Zimmer, and G. Weikum. Towards collaborative search in digital libraries using peer-to-peer technology. In *DELOS Workshop: Digital Library Architectures*, pages 61–72, 2004.
5. C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the seventeenth annual international ACM-SIGIR conference on research and development in information retrieval*. Springer-Verlag, 1994.
6. J. Callan. Distributed information retrieval. *Advances in information retrieval, Kluwer Academic Publishers.*, pages 127–150, 2000.
7. S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, San Francisco, 2002.
8. F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. Technical Report DCS-TR-487, Rutgers University, Sept. 2002.
9. R. Fagin. Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci.*, 58(1):83–99, 1999.
10. J. Li, B. Loo, J. Hellerstein, F. Kaashoek, D. Karger, and R. Morris. On the feasibility of peer-to-peer web indexing and search. In *In 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.
11. J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *Proceedings of CIKM03*, pages 199–206. ACM Press, 2003.
12. J. Luxenburger and G. Weikum. Query-log based authority analysis for web information search. In *WISE04*, 2004.
13. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM 2001*, pages 161–172. ACM Press, 2001.
14. P. Reynolds and A. Vahdat. Efficient peer-to-peer keyword searching. In *Proceedings of International Middleware Conference*, pages 21–40, June 2003.
15. A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.
16. J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000.
17. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM 2001*, pages 149–160. ACM Press, 2001.
18. T. Suel, C. Mathur, J. Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasunderam. Odissea: A peer-to-peer architecture for scalable web search and information retrieval. Technical report, Polytechnic Univ., 2003.
19. Y. Wang, L. Galanis, and D. J. de Witt. Galanx: An efficient peer-to-peer search engine system. *Available at http://www.cs.wisc.edu/ yuanwang*.
20. C. Yu, W. Meng, K.-L. Liu, W. Wu, and N. Rishe. Efficient and effective metasearch for a large number of text databases. In *Proceedings of CIKM99*, pages 217–224. ACM Press, 1999.

DIRECT: a Distributed Tool for Information Retrieval Evaluation Campaigns

Giorgio Maria Di Nunzio and Nicola Ferro

Department of Information Engineering – University of Padua
Via Gradenigo, 6/b – 35131 Padova – Italy
{dinunzio, nicola.ferro}@dei.unipd.it

Abstract. In this paper we describe the architecture of DIRECT, a Distributed Information Retrieval Evaluation Campaign Tool, which is an innovative system for managing evaluation campaigns. In particular, DIRECT deals with the evaluation of the information access and extraction components of a *Digital Library Management System (DLMS)*.

1 Introduction

Digital Library Management Systems (DLMSs) generally manage collections of multi-media digitalized data and include components that perform the storage, access, retrieval, and analysis of the collections of data. The evaluation of DLMSs is a non trivial issue that should cover different aspects, such as: the DLMS architecture, the DLMS information access and extraction capabilities, the management of multimedia content, the interaction with users, and so on [1]. We are interested in the evaluation aspects concerned with the information access and extraction components of a DLMS [2]; this interest ranges from measuring and quantifying the performances of the information access and extraction components of a DLMS to designing and developing an architecture capable of supporting this kind of evaluation in the context of DLMSs.

The paper is organized as follows: Section 2 presents the methodologies and the issues concerned with the evaluation of the information access components of a DLMS and discusses the motivations and the objectives of our system; Section 3 presents the architecture and the functionalities of our system; finally, Section 4 draws some conclusions.

2 Evaluation Issues for the Information Access Components of a DLMS

Today, this type of evaluation is carried out in important international evaluation forums which bring research groups together, provide them with the means for measuring the performances of their systems, discuss and compare their work. The most important forums for *Information Retrieval System (IRS)* are: *Text REtrieval Conference (TREC)*[1], *Cross Language Information Retrieval (CLIR)*[2],

[1] http://trec.nist.gov/
[2] http://clef.isti.cnr.it/

NII-NACSIS Test Collection for IR Systems (NTCIR)[3], and *INitiative for the Evaluation of XML Retrieval (INEX)*[4]. A wide range of questions are covered by these forums like the quality of the information retrieved, the access of multilingual collections of documents, the retrieval of structured documents and the access to asian-language collections. In general, these evaluation campaigns follow the Cranfield paradigm [3] giving the participants one or more test-bed collections, a set of tasks to be performed, and a method by which evaluating performances of their systems with respect to the defined collections and tasks.

A drawback of the approach followed during these evaluation forums is that huge chunks of textual files are shifted from side to side. Document collections usually reside on a single high-loaded server where all participants connect concurrently in a very short limited time in order to download the collections needed to carry out the experiments. The experimental results provided by participants to organizers usually consist of large text files, containing lists of retrieved documents together with their rank and score, that are numerical data. The performance figures computed by organizers and returned to participants consist of text files full of numerical data; in particular, the presentation format adopted by TrecEval[5], the de-facto standard tool for computing the performance figures, is not very suitable for direct processing by a computer program, since it is tailored to be human-readable. These file transfers often requires a mass-mailing between participants and organizers in order to acknowledge the receipt of the files or to correct errors. Moreover, if the performance figures are to be accessed in order to further process them, a lot of textual parsing is needed to transform chunks of text into numerical values, a process which is prone to errors.

Another drawback is that while the performance of IRS are measured by means of traditional IR performance indicators, the analysis of the significance of this results is rarely performed by participants although statistical analysis is a fundamental step in the experimental evaluation, as pointed out by [4]. We identify two main reasons for this: first, the analysis of the whole set of runs submitted is possible only by organizers that collect all the runs, and replicating experiments of other research groups is seldom possible for participants. Second, statistical tools are not easy to handle and the possibility to have non coherent results is high when participants make use of different tools. Moreover, the statistical analysis is burdened by all the textual parsing needed to transform chunks of text into numerical and processable data.

In this paper we want to tackle the problem of a new approach to evaluation campaigns, able to take into account the distributed nature of the entities involved during an evaluation campaign: data collections may reside on different servers, participants are scattered around the world, as well as assessors and organizers. Moreover, DLMSs themselves are distributed systems where the services under evaluation can be developed according to different architectural paradigms, such as *Web Services (WS)*, *Peer-To-Peer (P2P)*, and Grid. Finally,

[3] `http://research.nii.ac.jp/ntcir/index-en.html`
[4] `http://inex.is.informatik.uni-duisburg.de/`
[5] `ftp://ftp.cs.cornell.edu/pub/smart/`

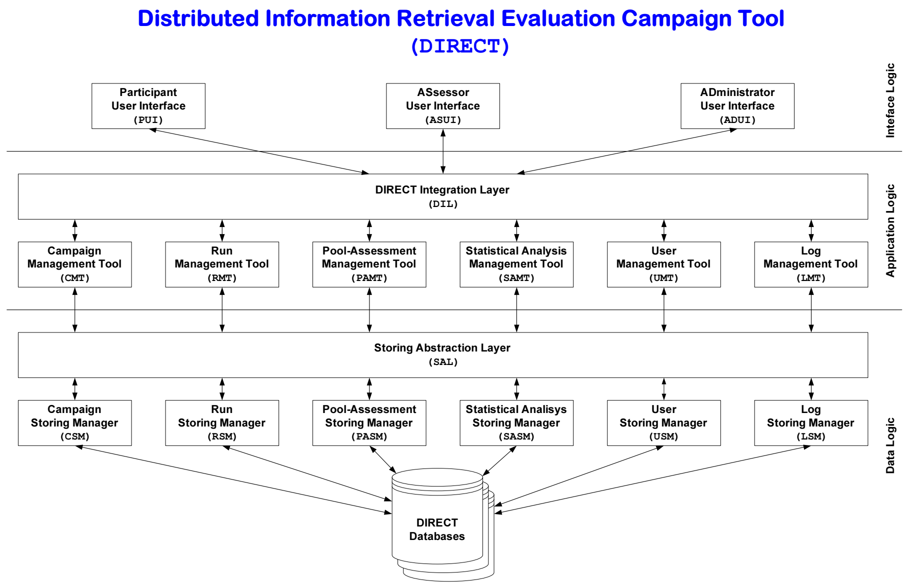

Distributed Information Retrieval Evaluation Campaign Tool
(DIRECT)
Participant User Interface (PUI)
ASsessor User Interface (ASUI)
ADministrator User Interface (ADUI)
Inteface Logic
DIRECT Integration Layer (DIL)
Application Logic
Campaign Management Tool (CMT)
Run Management Tool (RMT)
Pool-Assessment Management Tool (PAMT)
Statistical Analysis Management Tool (SAMT)
User Management Tool (UMT)
Log Management Tool (LMT)
Storing Abstraction Layer (SAL)
Campaign Storing Manager (CSM)
Run Storing Manager (RSM)
Pool-Assessment Storing Manager (PASM)
Statistical Analisys Storing Manager (SASM)
User Storing Manager (USM)
Log Storing Manager (LSM)
Data Logic
DIRECT Databases

Fig. 1. DIRECT Architecture

another innovative aspect of our approach is to provide participants with a uniform way of performing statistical analysis on their results. In this way, not only participants benefit from standard experimental collections but also they may exploit standard tools for the analysis of the experimental results. This approach, that makes the analysis and assessment of experimental results comparable, is quite innovative since up to now participants employed tools built on their own in order to analyze experimental results, making such analyzes much more diffucult to compare.

An innovative system named *Distributed Information Retrieval Evaluation Campaign Tool (DIRECT)* is being designed and developed to give an alternative to the management of data of these evaluation forums with the aim of integrating the activities among the different entities (both of an evaluation campaign and a DLMS) and giving the tools to make the activities themselves more interactive. The goal will be to create a unified view of this kind of evaluation forums and to propose an innovative architecture able to provide dedicated services and tools to make available data and documents. In particular, the evaluation of information access components of a DLMS will not be calculated by means of standard IR measures only, but also with an integrated tools for statistical analysis available to all participants of evaluation forums.

Since we are are going to provide and manage the technical infrastructure, both hardware and software, for the *Cross-Language Evaluation Forum (CLEF)* 2005 ongoing evaluation campaign, the possibility of testing and evaluating the DIRECT system in real settings will be exploited.

3 DIRECT Architecture and Functionalities

Figure 1 shows the architecture of DIRECT. It consists of three layers – data, application and interface logic layers – which allow us to achieve a better modularity and to properly describe the behavior of DIRECT by isolating specific functionalities at the proper layer. Moreover, this decomposition makes it possible to clearly define the functioning of DIRECT by means of communication paths that connect the different components. In this way, the behavior of the system is designed in a modular and extensible way.

In the following, we briefly describe the architecture shown in Fig. 1, from bottom to top:

Data Logic The data logic layer deals with the actual storage of the different information objects coming from the upper layers. There is a set of "storing managers" which translate the requests that arrive from the upper layers into *Structured Query Language (SQL)* statements to operate on the underlying *DataBase Management Systems (DBMSs)*. The heart of the data logic is an *Entity–Relationship (ER)* schema that is designed to fulfill the requirements to manage a complex evaluation forums like the ones presented in Section 2. Note that, due to huge quantity of data to be managed, it may be necessary to split the underlying database across different DBMSs, thus dealing with a distributed database. Finally, on top of the various "storing managers" there is the *Storing Abstraction Layer (SAL)* which hides the details about the storage management from the upper layers. In this way, the addition of a new "storing manager" is totally transparent for the upper layers.

Application Logic The application logic layer deals with the flow of operations within DIRECT. It provides a set of tools capable of managing high-level tasks. For example, the *Statistical Analysis Management Tool (SAMT)* offers the functionalities needed to conduct a statistical analysis on a set of runs; conducting a statistical analysis involves, in the data logic layer, both the *Run Storing Manager (RSM)* to have access to the experimental data, and the *Statistical Analysis Storing Manager (SASM)* to store the results of such analysis. Finally, the *DIRECT Integration Layer (DIL)* provides the interface logic layer with a uniform and integrated access to the various tools. As we noticed in the case of the SAL, thanks to the DIL also the addition of new tools is transparent for the interface logic layer.

Interface Logic It is the highest level of the architecture, and it is the access point for the user to interact with the system. It provides specialised *User Interfaces (UIs)* for different types of users, that are the participants, the assessors, and the administrators of DIRECT.

4 Discussion

We introduced DIRECT, a system for managing information retrieval evaluation campaigns, and described its architecture. DIRECT can be implemented according to different architectural paradigms: for example, the data logic layer can be implemented as a network of P2P "storing managers" in order to distribute the databases. On the other hand, the various tools of the application logic layer could be made available as WSs in order to easily access them and to compose them, if necessary. In conclusion, DIRECT not only allows for managing evaluation campaigns in a distributed manner but it is also a distributed tool itself.

DIRECT can be considered a *scientific databases* manager since it stores scientific data and makes it possible the analysis of the results of computations and data itself. It can become also a *curated database* manager if we partner it with services for annotating its contents, as those described in [5,6], in order to allow users to cooperate and discuss about the performances of different DLMSs in accessing information. In this context, *data provenance* [7] becomes a relevant issues and annotations can be further exploited for data provenance purposes, as described in [8,9].

The main goal of DIRECT is to model the data of the domain of interest (the evaluation forums) and to make available integrated services to operate on this data. The modelling is focussed on the problems explained in Sect.1 in order to efficiently manage the well defined tasks of accessing, (down-)loading, evaluating, submitting data during evaluation forums. In future, the infrastructure of DIRECT can be thought embedded as a part of a more complex system designed by means of scientific workflow management systems like Kepler/PtolemyII[6] [10].

DIRECT is going to be used and tested in the context of the ongoing CLEF evaluation campaign and the outcomes of this test can drive the further development and refinement of it.

Acknowledgements

The work is partially supported by the DELOS Network of Excellence on Digital Libraries, as part of the Information Society Technologies (IST) Program of the European Commission (Contract G038-507618).

References

1. Fuhr, N., Hansen, P., Micsik, A., Sølvberg, I.: Digital Libraries: A Generic Classification Scheme. In Constantopoulos, P., Sølvberg, I.T., eds.: Proc. 5th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2001), Lecture Notes in Computer Science (LNCS) 2163, Springer, Heidelberg, Germany (2001) 187–199

[6] `www.kepler-project.org`

2. Agosti, M., Di Nunzio, G.M., Ferro, N.: Evaluation of a Digital Library System. In Agosti, M., Fuhr, N., eds.: Notes of the DELOS WP7 Workshop on the Evaluation of Digital Libraries, `http://dlib.ionio.gr/wp7/workshop2004_program.html` (2004) 73–78
3. Cleverdon, C.W.: The Cranfield Tests on Index Languages Devices. In Spack Jones, K., Willett, P., eds.: Readings in Information Retrieval, Morgan Kaufmann Publisher, Inc., San Francisco, California, USA (1997) 47–60
4. Hull, D.: Using Statistical Testing in the Evaluation of Retrieval Experiments. In Korfhage, R., Rasmussen, E., Willett, P., eds.: Proc. 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1993), ACM Press, New York, USA (1993) 329–338
5. Agosti, M., Ferro, N.: An Information Service Architecture for Annotations. In Agosti, M., Schek, H.J., Türker, C., eds.: Digital Library Architectures: Peer-to-Peer, Grid, and Service-Orientation, Pre-proceedings of the 6th Thematic Workshop of the EU Network of Excellence DELOS, Edizioni Libreria Progetto, Padova, Italy (2004) 115–126
6. Agosti, M., Ferro, N., Frommholz, I., Thiel, U.: Annotations in Digital Libraries and Collaboratories – Facets, Models and Usage. In Heery, R., Lyon, L., eds.: Proc. 8th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2004), Lecture Notes in Computer Science (LNCS) 3232, Springer, Heidelberg, Germany (2004) 244–255
7. Buneman, P., Khanna, S., Tan, W.C.: Why and Where: A Characterization of Data Provenance. In Van den Bussche, J., Vianu, V., eds.: Proc. 8th International Conference on Database Theory (ICDT 2001), Lecture Notes in Computer Science (LNCS) 1973, Springer, Heidelberg, Germany (2001) 316–330
8. Buneman, P., Khanna, S., Tan, W.C.: On Propagation of Deletions and Annotations Through Views. In Abiteboul, S., Kolaitis, P.G., Popa, L., eds.: Proc. 21st ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems (PODS 2002), ACM Press, New York, USA (2002) 150–158
9. Bhagwat, D., Chiticariu, L., Tan, W.C., Vijayvargiya, G.: An Annotation Management System for Relational Databases. In Nascimento, M.A., Özsu, M.T., Kossmann, D., Miller, R.J., Blakeley, J.A., Schiefer, K.B., eds.: Proc. 30th International Conference on Very Large Data Bases (VLDB 2004), Morgan Kaufmann (2004) 900–911
10. Ludscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger-Frank, E., Jones, M., Lee, E., Tao, J., Zhao, Y.: Scientific Workflow Management and the Kepler System. Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows (2005) (in print)

ρ-index – An Index for Graph Structured Data

Stanislav Bartoň and Pavel Zezula

Faculty of Informatics, Masaryk University, Brno, Czech Republic
{xbarton, zezula}@fi.muni.cz

Abstract. The effort described in this paper introduces an indexing structure for path search in the graph structured data called ρ-index. It is based on a graph segmentation $S(G)$ that is meant to represent the indexed graph G in a simpler manor yet having similar properties as the graph G had. This is achieved using graph transformations and a special type of a matrix used to represent the transformed graph.

1 Introduction

In the context of the Semantic Web, ρ-operators are proposed in [1] as a mean to explore complex relationships [3] between entities. The problem of searching for the complex relationships can be modeled as the process of searching paths in a graph where entities represent vertices and edges the relationships between them. The notion of complex relationships can be also identified in bibliographic digital libraries, where entities could represent publications and the relationship can represent references or citations between them.

As proposed in [1], we recognize two kinds of complex relationships. The first one is represented by *a path* lying between two inspected vertices. Speaking in terms of publications this means that one publication indirectly cites or references the other publication – a chain of publications can be built so that one cites another. The second type of complex relationship is *a connection* between two inspected vertices. This symbolizes a fact that the two inspected publications indirectly cite one common publication, see Fig. 1 for an example of this kind of complex relationship.

The knowledge about complex relationships among publications can be used for example for ranking the result of the search for publications. Another use case can be an automated recommendation of publications based on the preferred set of publications. For that reason, this paper presents an indexing technique called the ρ-index that enables efficient discovery of complex relationships in large collections of graph structured data.

2 Motivation

The graph theory proved that a very handy representation of a directed graph is its adjacency matrix because using matrix algebra we can comfortably study the graph's properties. For instance, if the adjacency matrix is powered by two,

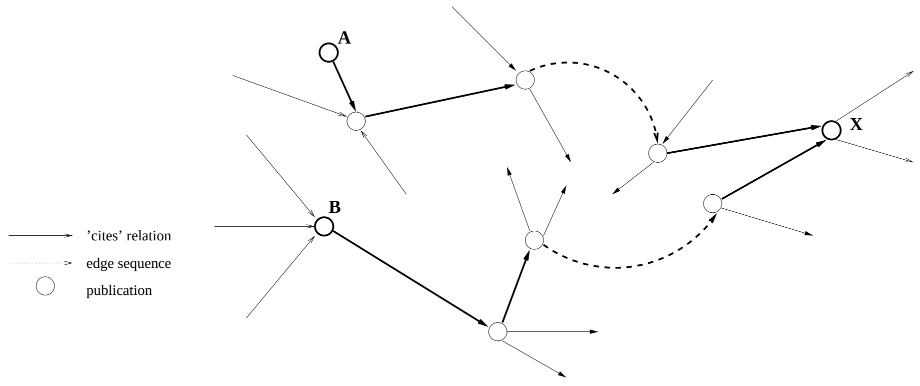


Fig. 1. An example of a connection between vertices A and B. Two paths originated in A and B connected in a common vertex X.

each field in the resulting matrix contains a number of paths of length two lying between every pair of vertices in the original graph. If the computation continued, the result would contain amounts of all paths of an arbitrary length. Moreover, with a slight modification of the matrix that is introduced in section 4.2 we would get not just the amounts of paths but the paths themselves.

The main difficulty of a matrix representation of a graph is that its use is limited to fairly small graphs, because the matrix grows in the quadratic space and the multiplication operation on matrices has even cubic time complexity. Therefore, we introduce graph transformations to enable the use of the matrix approach to graphs of arbitrary size.

3 Theoretical background

The proposed indexing structure is built upon the theoretical base of the graph theory and following theses. The necessary background definitions can be found in Appendix A to this paper. The corner stone of this work is *a graph segmentation. A graph segment* is very similar to a graph's subgraph. The main difference is that the segment also can contain edges that have only one vertex in the proper segment. The graph segmentation is then a division of the original graph into segments where is true that no vertex belongs to two different segments and that the whole graph is segmented – all vertices and edges are present in the segmentation. The purpose of the graph segmentation $S(G)$ is a simplified representation of a graph G by a segment graph $SG(G)$ that has similar properties that G had. Basically, following a path p in G implies following a path in $SG(G)$ represented by *a proper segment sequence* of p.

Lemma 1. *If a graph $G = (V, E)$ has a segmentation $S(G)$ that forms a graph $SG(G)$, any path $p = (v_1e_1v_2e_2 \dots e_nv_{n+1})$ in G can be represented by its proper segment sequence in $S(G)$ and this representation is unique.*

Proof. A segment sequence is another representation of a path in $SG(G)$. In fact, $(S_1 \dots S_l)$ is a simplified representation of $(S_1 h_1 S_2 h_2 \dots S_{l-1} h_{l-1} S_l)$. Thus, we show that any path in G can be transformed into a path in $SG(G)$ that actually represents a segment sequence which is the proper segment sequence for this path.

From the definition of $S(G)$ we have that each vertex in G belongs exactly to one segment of $S(G)$. Therefore, we take the path p in G and rename the vertices by the segments they belong to.

$p = (v_1 e_1 v_2 e_2 \dots e_n v_{n+1}) \longrightarrow (S_1 e_1 S_2 e_2 \dots e_n S_{n+1})$

If $S_i = S_{i+1}$ then e_i is not a border edge, therefore we omit the part (e_i, S_{i+1}) from the transformed path. We repeat this step until $S_i \neq S_{i+1}$ is true.

According to the definition of $SG(G)$ we drew an edge $h = (S_1, S_2)$ in $SG(G)$ whenever $EDGES\_OUT(S_1) \cap EDGES\_IN(S_2) \neq \emptyset$. Presence of e_i between S_i and S_{i+1} in $(S_1 e_1 S_2 e_2 \dots e_n S_{n+1})$ implies the presence of such edge in G connecting two vertices from the particular segments where $S_i \neq S_{i+1}$ which implies that $EDGES\_OUT(S_i) \cap EDGES\_IN(S_{i+1}) \neq \emptyset$ and therefore exists an edge h in $SG(G)$ from S_i to S_{i+1}.

Now we have $(S_1 e_{i_1} S_{i_1} e_{i_2} \dots e_{i_{l-1}} S_l)$, so we replace all e_i by the respective edges h_j from $SG(G)$. The result is a correct path $(S_1 h_1 S_{i_1} h_2 \dots h_l S_l)$ in $SG(G)$ that represents a proper segment sequence $(S_1 \dots S_l)$ for the path p in G.

The uniqueness of the proper segment sequence of a path p results from the definition of segmentation $S(G)$ because no vertex in V can be included in two different segments. Thus, the chain of segment labels is unique for any path in G.

Lemma 2. *If a graph $G = (V, E)$ has a segmentation $S(G)$ that forms a graph $SG(G)$, the length of a path in $SG(G)$ that represents a proper segment sequence of a path p is always less than or equal to the length of p.*

Proof. From Lemma 1 we know that each path in G has its proper segment sequence representation. From the proof of that lemma we know that during the transformation of the path in G to a path in $SG(G)$ we omit zero or more edges. Also some edges from G are replaced by edges from $SG(G)$ but always one edge by another. This implies that the resulting path in $SG(G)$ can be at most of the same length as the path in G.

4 Overview of the ρ-index

The main idea of the approach introduced in this paper is to identify certain units in the indexed graph such that when replaced by single vertices, they would form a new smaller graph that would be easier to navigate, but yet having the same properties as the original graph had. The aim is to enable the use of the matrix approach on $SG(G)$ while it is not possible to use it on G. And because $SG(G)$ is also a regular graph it can be again segmented.

4.1 Proposed graph transformations

The first graph transformation used is *a graph to a forest of trees* transformation. The result of this transformation is a set of trees together with a set of transitions among those trees. A lot of indexing techniques for trees have been developed for efficient navigation inside a tree. We have chosen the tree signatures [4]. They enable fast navigation inside a tree using simple and cheap operation – a comparison of preorder and postorder ranks of nodes in the particular tree. The ranks are represented by integer numbers thus the comparison of the particular ranks takes $\mathcal{O}(1)$ time.

If we take a closer look on the result of this transformation we see that the acquired trees form vertices and transitions among them edges in a new graph. This new graph represents the original graph but in a simplified way. Certainly, any path that was in the original graph is also in the new graph and vice versa.

If the size of the newly acquired graph is small enough to create the matrix representation of it, the job is done. The index would be then composed of the tree signatures and the matrix describing all paths among those trees. However, we would like to index a graph of an arbitrary size. An obvious idea is to use this graph transformation recursively onto the newly acquired graph as long as we get a graph of a desired size. But an evaluation of the recursive application of this transformation in [2] showed that after few applications this method does not lead to a significant reduction of the amount of nodes in the new graph.

Therefore, another graph transformation has to be used to lower the amount of vertices in the new graph. The transformation that we have chosen for this is *vertex clustering*. It reduces the amount of vertices in a graph by collapsing subgraphs (segments) into single vertices. Also in the new graph acquired by this transformation is true that any path in the new graph is contained in the original one and vice versa. This graph transformation represents the graph segmentation, thus, the new graph is a segment graph of the original graph.

The reason why the transformation of graph to forest of trees is used is to make more dense graph out of a sparse graph. The tree signatures can be applied to tree of any size. This kind of transformation is used only once as a first step in the process. In all following steps only the vertex clustering is used because the number of vertices in the transformed graph does not decrease in a linear way but rather converges to a certain limit.

4.2 Adjacency matrix of paths

A path type adjacency matrix is a modification of a usual adjacency matrix. It is designed to represent a graph in a path oriented way. Initially, each field of our matrix contains a path consisting of a single edge whenever there exists such an edge between two vertices in the graph. The convenience it presents over the usual adjacency matrix is that after the transitive closure of the path type matrix is computed, the fields contain not just an amount of paths lying between any two vertices, but also the paths themselves. Naturally, the mathematical operations on numbers $+$ and $*$ are replaced by the respective operations on paths - union and concatenation.

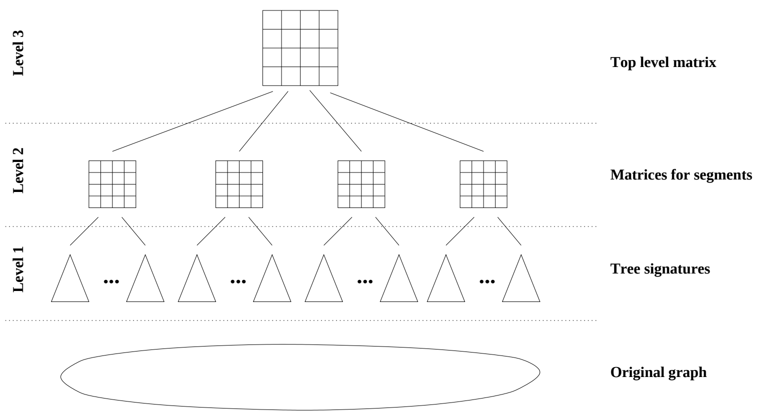


Fig. 2. A ρ-index consisting of three levels.

4.3 Outline of ρ-index's structure

Let present a brief example of a three level ρ-index visualized in Fig. 2. Firstly, the graph to a forest of trees transformation is applied to the graph that is indexed. Subsequently, a tree signature is created to each acquired tree. The new graph, where vertices represent trees and edges represent transitions among trees, is then decomposed by the vertex clustering transformation. For each collapsed subgraph, its path type adjacency matrix is created. After that a path type adjacency matrix is created to the newly acquired graph, where vertices represent collapsed subgraphs (segments).

Hence, the ρ-index has the following structure. On the lowest level, there are tree signatures of trees obtained by the first graph transformation. Above that is a set of path type matrices describing each collapsed subgraph. And at the topmost level is a single path type matrix used to navigate among the subgraphs. This particular example is visualized in Fig. 2.

5 Preliminary experimental evaluation

An experimental implementation of the ρ-index was built to evaluate its properties. The measurable aspects are the time necessary to build the ρ-index and the time consumed to discover the relationships. Furthermore, a proportion of accessible vertices from the inspected vertex in the indexed graph to the amount of actually accessed vertices in the ρ-index is measured. Also a total number of accesses to vertices in the ρ-index is recorded.

The both graphs used to evaluate the ρ-index properties are a part of the Open Directory Project[1] representing the connections among categories. They

[1] The Open Directory Project can be found at http://www.dmoz.org.

mostly differ in the size and density. We have run the tests on a usual desktop computer with a 3 GHz Pentium 4 processor and 2 gigabytes of RAM.

# of levels in ρ-index	3		3	
# of vertices	15 214		169 271	
# of edges	66 478		255 687	
Time to create (mins)	0:45		1:52	
# of accessible vertices	6419		66 065	
Resulting relationship	found	not found	found	not found
# of accessed vertices	462	130	17	11
# of accesses	162 492	5534	97	157
Time to find relationship (secs)	10	2	0.01	0.01

Table 1. Experimental evaluation of ρ-index.

The ρ-index was tested on two graphs that differed in the amount of vertices they contained. The proportions of both graphs are stated in Table 1. As the total number of edges in each graph proposes, the smaller graph is more dense than the bigger one. The evaluation consisted of a set of executions of a path relationship search. Thus, we used the index to retrieve all paths lying between the two inspected vertices. The result of this search was either the an empty set or all paths lying between the two vertices. During each execution the total number of accesses to vertices in the ρ-index was recorded to measure the efficiency of designed algorithms.

The number of accessible vertices indicates the amount of vertices that can be accessed in the indexed graph from the starting vertex. Number of accessed vertices represents an amount of actually accessed vertices in the ρ-index.

The experimental evaluation concludes that the sparser the graph is the better results we get from the ρ-index . Why is that true indicates the total number of accesses to vertices in the ρ-index . We found out that not every segment sequence that is a product of the transitive closure computation of $SG(G)$ does represent some path on the lower level (in G). And this happens more often in the denser graph causing a huge amounts of these false segment sequences to be checked by the search algorithm.

6 Concluding Remarks & Future Work

During the ρ-index implementation, some further modifications had to be done to the theoretic design of the ρ-index . The theoretic base of the ρ-index counted on computation and storage of all possible paths along the building phase of the index. Nonetheless, huge amounts of paths computed at the higher levels of the structure turn to be a dead end during the retrieval of a real path between two vertices in the original graph. Therefore, limitations concerning the maximal amount of paths stored in each path type matrix field and a maximal iteration step of the transitive closure computation of path type matrix were set during the ρ-index creation phase. These limitations have impact on the quality of the response retrieved using the ρ-index, although they enable the user with the control over the maximal length of an indexed path and a maximal number of paths indexed between two vertices. Albeit, these do not limit the use of ρ-index, since in many cases the user's concern is limited to the most relevant relationships.

As the experimental evaluation of the ρ-index implied, the future work will mainly focus on the design improvement of the ρ-index . One of the possible ways is to control the undesired storage of segment sequences that do not represent any path on lower levels. Another direction we draw our attention will be to index paths and connections only to a certain length and to a certain amount between any two vertices. To achieve that we would like to introduce weights of vertices to promote their importance.

References

1. Kemafor Anyanwu and Amit Sheth. ρ-queries: enabling querying for semantic associations on the semantic web. In *Proceedings of the twelfth international conference on World Wide Web*, pages 690–699. ACM Press, 2003.
2. Stanislav Bartoň. Indexing structure for discovering relationships in RDF graph recursively applying tree transformation. In *Proceedings of the Semantic Web Workshop at 27th Annual International ACM SIGIR Conference*, pages 58–68, 2004.
3. Sanjeev Thacker, Amit Sheth, and Shuchi Patel. Complex relationships for the semantic web. In D. Fensel, J. Hendler, H. Liebermann, and W. Wahlster, editors, *Spinning the Semantic Web*. MIT Press, 2002.
4. Pavel Zezula, Giuseppe Amato, Franca Debole, and Fausto Rabitti. Tree signatures for XML querying and navigation. *Lecture Notes in Computer Science*, 2824:149–163, 2003.

Appendix A

Definitions

Definitions necessary to state the theoretical background of the approach introduced in this paper.

1. Vertices $V = \{v_1, ...v_n\}$
2. Edges $E = \{e_1, ...e_m\}, E = V \times V, e_i = (v, w), v, w \in V$
3. Graph $G = (V, E)$
4. Initial vertex of an edge e: $LEFT\_VTX(e) = v_1 \Leftrightarrow e = (v_1, v_2)$
5. Terminal vertex of an edge e: $RIGHT\_VTX(e) = v_2 \Leftrightarrow e = (v_1, v_2)$
6. *Segment* S in graph G : $S = (V_S, E_S)$: $V_S \subseteq V \ \wedge \ V_E = \{e \in E \mid RIGHT\_VTX(e) \in V_S \ \vee LEFT\_VTX(e) \in V_S\}$

7. *Segmentation* $S(G) = \{S | S \textit{ is a segment of } G\} \wedge \bigcap_{S \in S(G)} V_S = \emptyset \wedge \bigcup_{S \in S(G)} V_S = V$
8. EDGES_OUT(S) $= \{e | e \in E_S \wedge LEFT\_VTX(e) \in V_S \wedge RIGHT\_VTX(e) \notin V_S\}$
9. EDGES_IN(S) $= \{e | e \in E_S \wedge RIGHT\_VTX(e) \in V_S \wedge LEFT\_VTX(e) \notin V_S\}$
10. Sequence of segments $(S_1 \dots S_l) = S_1, \dots S_l \in S(G),\ 1 \leq i \leq l-1 : EDGES\_OUT(S_i) \cap EDGES\_IN(S_{i+1}) \neq \emptyset$
11. Acyclic sequence of segments $(S_1 \dots S_l)$ is a sequence of segments where: $1 \leq i \neq j \leq l-1 : S_i = S_j \Rightarrow S_{i+1} \neq S_{j+1}$
12. Acyclic path $p = (v_1 e_1 v_2 e_2 \dots e_n v_{n+1})$ in $G : 1 \leq i \leq n, 1 \leq j \leq n+1, i \neq j : e_i \in E \ \wedge\ v_i, v_j \in V \ \wedge\ v_i = LEFT\_VTX(e_i) \ \wedge\ v_{i+1} = RIGHT\_VTX(e_i) \ \wedge\ v_i \neq v_j$
13. Simplified path notation: instead of $(v_1 e_1 v_2 e_2 \dots e_n v_{n+1})$ we sometimes use $(e_1 e_2 \dots e_n)$.
14. *Connecting path* $p = (e_1 e_2 \dots e_n)$ in a segment sequence $(S_1 \dots S_l)$: $p \in (S_1 \dots S_l) : e_1 \in \{EDGES\_OUT(S_1) \cap EDGES\_IN(S_2)\} \wedge e_n \in \{EDGES\_OUT(S_{l-1}) \ \cap\ EDGES\_IN(S_l)\} \ \wedge\ \exists i_2, i_3, \dots i_{l-1} : 1 < i_2 < i_3 < \dots < i_{l-1} < n : \{e_2, \dots e_{i_2}\} \subseteq E_{S_2} \wedge \{e_{i_2}, \dots e_{i_3}\} \subseteq E_{S_3} \wedge \dots \wedge \{e_{i_{l-2}}, \dots e_{i_{l-1}}\} \subseteq E_{S_{l-1}}$
15. *Proper segment sequence* for a path $p = (v_1 e_1 v_2 e_2 \dots e_n v_{n+1}) : S(p) = (S_1 \dots S_l) : S(p) \textit{ is a segment sequence } \wedge 1 \leq i_1 < i_2 < \dots < i_l \leq n+1 : \{v_1, \dots v_{i_1}\} \subseteq V_{S_1} \wedge \{v_{i_1}, \dots v_{i_2}\} \subseteq V_{S_2} \wedge \dots \wedge \{v_{i_l}, \dots v_{n+1}\} \subseteq V_{S_l}$
16. Segment graph of G: $SG(G) = (S(G),\ X),\ X = \{h | h = (S_i, S_j) \Leftrightarrow 1 \leq i, j \leq k \wedge EDGES\_OUT(S_i) \cap EDGES\_IN(S_j) \neq \emptyset\}$

Efficient and Effective Matching of Compound Patient Records

Michael Springmann[1], Sören Balko[2], and Hans–Jörg Schek[12]

[1] Institute for Information Systems,
University for Health Sciences, Medical Informatics and Technology (UMIT)
EWZ 1, A-6060 Hall i.T., Austria
{michael.springmann,hans-joerg.schek}@umit.at
[2] Database Research Group
Swiss Federal Institute of Technology (ETH)
ETH Zentrum, CH-8092 Zurich, Switzerland
{sbalko,schek}@inf.ethz.ch

Abstract. Patient records form an important cornerstone of nowadays medical information systems. Technically, they represent complex multimedia documents that comprise temporally interrelated elements of various media types. The collection of patient records of a health care institution forms a digital library, with access limited to this organization. Efficient similarity search in large patient record databases is an important foundation for a number of highly rewarding analysis operations. Therefore we propose a novel approach to similarity retrieval of complex multimedia documents, in general, and patient records, more specifically. That is, we adapt and extend a retrieval algorithm that is known to work efficiently for region matching of images. Specifically, we introduce efficient and effective domain-specific matching of patient records, that (1) exclusively matches elements of identical media types (like *x-ray images*) with (2) dedicated measures (like *nearest neighbor on texture features*) and (3) only establishes semantically meaningful matches (like *x-rays of same organs in identical orientation*). Our approach is not limited to the medical domain and can be used for any large-scale digital library of compound multimedia documents.

1 Introduction

This section provides an overview on structure and content of patient records and motivates our contribution on fast matching in large collections of patient records.

1.1 Motivation

Electronic patient records have become an important patient-centered entity in health care management. They represent complex documents that comprise a wide diversity of patient-related information, like basic administrative data (patient's name, address, birthday, name of physicians, hospitals, etc.), billing details (name of insurance, billing codes, etc.), and a variety of medical information (symptoms, diagnoses, treatment, medication, X-ray images, etc.). Most of these details are structurally and temporally interrelated, like a (1) particular treatment and (2) medication ordered from a (3) physician on an (4) established diagnosis from patient's (5) reported symptoms at a (6) certain date. Each item is of a dedicated media type, like structured alphanumeric data (e.g. patient name, address, billing information, laboratory values, documentation codes, etc.), semi-structured text (symptoms description, physician's notes, etc.), images (X-ray, CT, MRT, images for documentation in dermatology, etc.), video (endoscopy, sonography, etc.), time series (cardiograms, EEG, respiration, pulse rate, blood pressure, etc.), and possibly others, like DNA sequence data.

State-of-the-art medical information systems aggregate massive amounts of data in patient records. Its exploitation, however, is mainly restricted to accounting and billing purposes, leaving aside most medical information. In particular, similarity search based on complex document matching is out of scope for today's medical information systems.

1.2 Contribution

As stated in [1] there are many application fields in medicine for content-based retrieval methods. These application fields can roughly be divided into the three domains teaching, research and diagnosis. We believe that efficient and effective matching of patient records forms the rewarding foundation for a large application variety. Data mining applications that rest upon our patient record matching approach will foster enhanced therapy, in general. Specifically, our proposal allows for effective comparison of similar cases to (1) shorten necessary treatments, (2) improve diagnoses, and (3) guide the proper medication to improve patients' well-being and reduce cost.

Precisely, we introduce a novel matching algorithm that tackles two issues concurrently. On the one hand, we provide a query model that incorporates the particular composition of patient records. For the time being, we focus on the aggregate structure and different domains. On the other hand, our matching algorithms allows for efficient similarity search with interactive answering times while processing large document collections.

2 Compound Document Matching

Health records like any other compound multimedia documents comprise component objects. In contrast to many existing digital libraries, these compound documents do not consist of rather small numbers of objects, but aggregate many multimedia data items over years. For example, the Radiology Department of the University hospitals of Geneva produced medical images more than 12,000 per day in 2002 [1]. For a very rough estimation, when all 2197 beds of the hospital are in use, more than five images per patient are generated – on one single day of their life. The Picture Archiving and Communications System (PACS) of Tilak, a local health care provider in Tyrol with 2046 beds, has to cope even with 240 Gbytes of new data for a single day, which corresponds to 320.000 images [2]. The aim For this reason, analyzing patient records for similarity is more complex than comparing two short documents or web pages consisting of only a couple of paragraphs of text and images. But to illustrate the problem, this might be a good start.

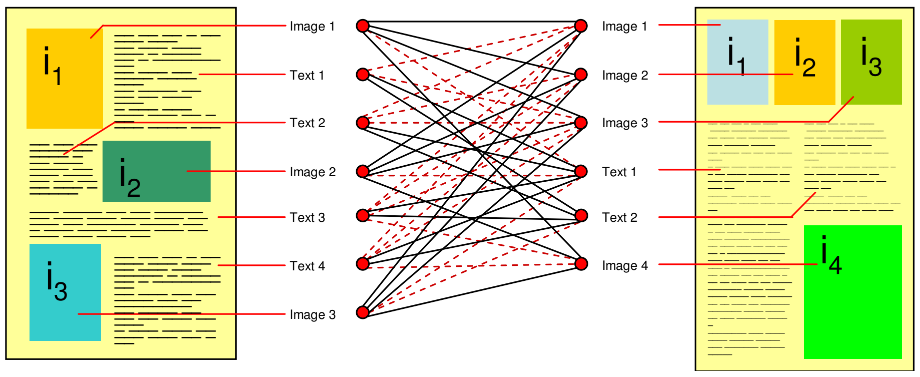


Fig. 1. Matching problem between two compound documents

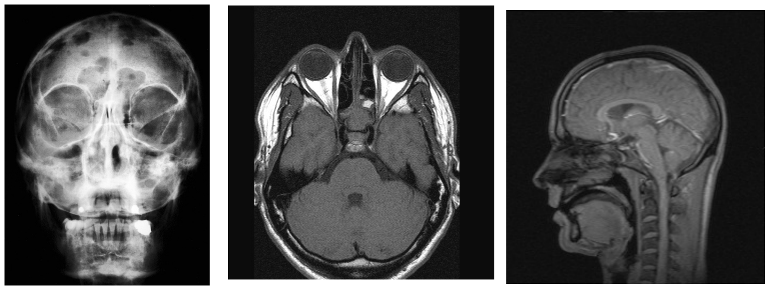
Similarity search on compound documents needs to establish matches of corresponding media objects. Finding these corresponding objects within large sets of objects is a computationally intensive task. A similar problem is prevalent in region based image retrieval (RBIR). That is, regions from two images must be matched to compute an overall similarity score.

In region matching like it is described in [3], images are partitioned into disjoint regions. Two images' overall similarity score is computed by finding best pairs of regions from those two images. This optimization problem of finding best pairs is a generalization of the so-called Assignment Problem. Using the Hungarian algorithm [4], the complexity of similarity computation for two region sets is $O(r^3)$ with r being the number of regions in both sets. From a query processing point of view, a cubic complexity is unsatisfactory, restricting the applicability to small data sets and few regions.

To perform this task more efficiently, [3] proposes a two-stage filter-and-refinement approach. Such an approach can reduce both I/O and CPU costs. E.g. in [5], such an approach is applied to reduce the number of file accesses needed to read feature vectors for content-based multimedia retrieval from disk. For complex queries, the costs of computation become more important and may exceed the costs for accessing the data from storage [6]. In this paper we focus on the CPU cost reduction.

The first stage of the approach yields a small number of candidate matches by computing computationally cheap, yet tight upper and lower bounds to the exact similarity score. The second stage invokes the exact distance function on the candidates. As the cost to compute lower and upper bounds is significantly lower than its exact counterpart, tremendous speed-ups can be achieved. In [3], authors report a factor of five in query processing time consumption.

Like images can be partitioned into regions, compound documents can be divided into their component objects. Figure 1 illustrates this for two documents, consisting of seven and six objects. Calculating the similarity of two compound documents c_1, c_2 must (1) establish matches between corresponding objects from both documents and (2) compute the document similarity out of the individual object similarities.



(a) X-Ray (b) MR 1 (c) MR 2

Fig. 2. Various medical images of human skulls

2.1 Finding Corresponding Objects

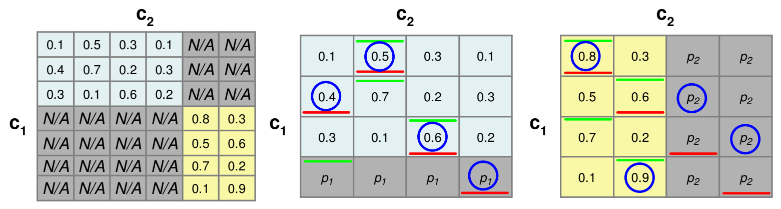
Every component object of c_1 should correspond to exactly one object of c_2 and vice versa. That is, we assign each object to exactly one other object, which is called a *complete match*. This is a restriction to the simple approach to match each object in c_1 straight forward to the object of c_2 which is most similar. This relaxed condition would allow assigning one object in c_2 twice or even to every object in c_1. In general, this is not the correct answer to a compound query document containing several distinct component objects. In [7] this is explained by requesting an image showing two tigers, but getting results showing only one.

For compound documents a similar problem may exist: If we query for a document containing one paragraph about one distinct topic, and another paragraph about one different topic, we may want to retrieve only documents handling the same two topics individually instead of mixing both topics in a single paragraph. With increasing number of component objects, the ability the model this distinction and retrieve only documents fulfilling the complete match criterion gains very much importance.

In contrast to region matching, compound documents may consist of objects from various domains. In figure 1 both documents have objects of the domains text and images. In order to form a complete match, we would need to assign one text to an image. But how can we be sure that the text *really* relates to the image? To be sure, we would need a similarity function comparing texts to images.

In general, no meaningful cross-domain similarity function exists, e.g. it is not feasible to match a cardiogram against a CT image of the lung. Even between objects of the same media type, a comparison can be problematic. If we think of medical imaging, DICOM files are generated via various techniques (CT, MRI, PET, etc.) which differ all in the way they represent tissues. Moreover, the screened body function and the spatial orientation of an image have an impact on the object domain, as well.

For example, all three images in figure 2 show the head of human patients which were stored in the DICOM file format. Traditional x-ray images show a 2D projection of whole 3D object, shown in 2(a), whereas newer technologies like computerized tomography (CT) and magnetic resonance imaging (MRI) allow to view discrete, two-dimensional cuts through the objects. But still, matching of CT and MRI according



(a) Disregarding Domains (b) Domain 1 (c) Domain 2

Fig. 3. Example similarity matrices for two documents c_1 and c_2 with 2 domains

to [8] can be performed only under certain conditions. And even if the same technology is used, like in 2(b) and 2(c), computing image similarity based on visual features may return poor result when the viewing perspective changes. One solution to the problem of coding the anatomical region, body orientation and used technology of medical images is suggested in [9], using the so-called IRMA code. It is a string of at most 13 characters {0–9,a–z} of the format TTTT–DDD–AAA–BBB, where T specifies the technology used, D the direction or body orientation, A the anatomical region, and B the biological system examined. This kind of knowledge can be used to provide similarity functions and judge their significance for application specific problems.

Therefore, the structure of a compound query document with regard to its component objects and their media types is one possible feature of the query. An improved approach might even define domains not solely on media types, but whether there exists a meaningful similarity measurement between two objects. The similarity between any two objects is computed using an appropriate similarity function for their domain and the result is normalized to the interval [0,1], such that 1 denotes most similar and 0 not similar.

We can build a similarity matrix where the rows depict objects from the compound (query) document c_1 and the columns the objects of the compared document c_2. The matrix is filled with the similarity score returned by the function for the respective objects. We do not limit ourselves to any particular similarity function. In fact, it is one of the major goals of this approach to be able to use an arbitrary existing similarity measure instead of developing new ones for each domain.

An example with 7 objects in c_1 and 6 in c_2 is shown in figure 3(a). The value *N/A* indicates that no meaningful match exists. Therefore, we split the matrix into the domain-specific matrices. Since it is not possible to find exact matches in non-quadratic matrices, we extend the split matrices until each has as many columns as rows and fill these new fields with so-called *penalty values* (cf. p_1 and p_2 in figure 3(b) and 3(c)). In this way, we may penalize (or reward) missing counterparts in c_2 to objects from c_1 (and vice versa). Notice that any complete match within a domain needs to assign the same number of penalty values. Hence, finding the best complete match of two distinct compound documents is orthogonal to the choice of penalty values. Therefore, the actual values of p_1 and p_2 do not affect the solution to our example. Their values gain only importance when ranking of the similarity between one query document and several other documents is performed and those other compound documents differ in the number of objects within at least one domain.

2.2 Document Similarity

Document matching solves an optimization problem, which is on finding the maximum overall similarity of two compound documents. That is, $sim(c_1, c_2)$ computes two documents' similarity out of their component object similarities. For the time being, we use the *unweighted average* of the objects' similarities, using a separate similarity matrix for each domain. By defining domains according to the availability of a meaningful similarity measure, we can guarantee that finding the best matches for each domain will also find the overall best match for the complete documents. This is due to the fact that every pair which is not covered within the restriction to domains can achieve only the similarity score *N/A* – which has to be treated as being worse than any other similarity score including any penalty value.

Performing similarity search using independent domains decreases the complexity of the computation significantly. If we would not distinguish between domains and would use a brute force approach to find the best match, we needed to compute the similarity for all $\frac{7!}{(7-6)!} = 5040$ possible matches in 3(a). Opposed to this, only $\frac{4!}{(4-3)!} = 24$ possible matches in 3(b) and $\frac{4!}{(4-2)!} = 12$ in 3(c) exist, and the optimum within each domain can be determined independently. Using the Hungarian algorithm reduces the complexity to $O(n^3)$, where without respect to domains $n = 7$ and with domains $n = max(4, 4)$.

The execution of the Hungarian algorithm is still computationally expensive for documents with many objects distributed among few domains. Following [3], we use a two-stage *filter-and-refinement*-algorithm. In a first stage, we find a small set of candidate documents using less expensive, approximative similarity functions. The second stage invokes the costly Hungarian algorithm on the remaining candidates.

To determine the set of candidates, we employ approximative, yet tight bounds on the document similarity. That is, we introduce an upper bound document similarity $sim_{ub}(c_1, c_2) \geq sim(c_1, c_2)$ and a lower bound similarity $sim_{lb}(c_1, c_2) \leq sim(c_1, c_2)$.

A valid upper bound to the document similarity is to simply choose a locally best match (maximum similarity) for each component object in the similarity matrix, as indicated in figure 3 by green bars above the numbers. In this way, we achieve upper bounds of $(0.5 + 0.7 + 0.6 + p_1)/4$ (domain 1) and $(0.8 + 0.6 + 0.7 + 0.9)/4$ (domain 2). Notice, that neither matches are complete, as two objects from c_1 match to the same object from c_2 (domain 1) with a similar situation in domain 2.

On the contrary, any complete match is a valid lower bound. To get a tight bound, we choose again the maximum similarity for each object match, but this time enforce a complete match. In the example, this is indicated by red bars below the numbers. The lower bound might return the overall best solution, but since it also uses only local maxima, it is not guaranteed to return the global maximum. In the example, the lower bound similarity is $(0.5 + 0.4 + 0.6 + p_1)/4$ (domain 1) and $(0.8 + 0.6 + p_2 + p_2)/4$ (domain 2).

```
FIND-MOST-SIMILAR-DOCUMENT($c_Q, C$)
    ▷ First phase: Find candidates
    ▷ Let heap be decreasingly ordered on upper bounds
1   $threshold \leftarrow 0$                          ▷ initialize to worst possible score
2   for each Compound Document $c_i \in C$
3       do
4           determine $sim_{ub}(c_Q, c_i)$
5           if $sim_{ub}(c_Q, c_i) \geq threshold$        ▷ $c_i$ is a candidate
6             then heap.push($sim_{ub}(c_Q, c_i), c_i$)   ▷ insert $c_i$ to queue of candidates
7                  determine $sim_{lb}(c_Q, c_i)$
8                  if $sim_{lb}(c_Q, c_i) > threshold$     ▷ $threshold$ needs to be updated
9                    then $threshold \leftarrow sim_{lb}(c_Q, c_i)$

    ▷ Second phase: Compute exact similarity
10  while (!heap.isEmpty())
11      do $(sim_{ub}, c) \leftarrow$ heap.pop()          ▷ retrieve next candidate
12          if $sim_{ub}$ is not the exact similarity
13            then determine $sim(c_Q, c)$                ▷ this calls the Hungarian algorithm
14                 heap.push($sim(c_Q, c), c$)            ▷ enqueue again with exact similarity
15            else return $c$                             ▷ $c$ is the most similar document to $c_Q$
```

Fig. 4. Filter-and-refinement algorithm

Using the Hungarian algorithm, we can compute the overall maximum for each domain. The result is indicated by blue circles around the similarity scores. For domain 1 it is the same as the lower bound, for domain 2 it is $(0.8 + p_2 + p_2 + 0.9)/4$.

To calculate the global document similarity we simply compute the average of the domains' similarities. This can be done with either similarity score: upper bound, lower bound, or exact value. For instance, $sim_{ub}(c_1, c_2)$ actually is the average of the upper bounds for all domains in c_1 and c_2.

The two-stage filter-and-refinement algorithm is shown in figure 4. c_Q denotes the query document and C the collection of all compound documents to which we want to compare c_Q. The filter-and-refinement algorithm uses a heap as a priority queue ordered by the evaluated similarity values.

In the first stage c_i is the current document to which c_Q is compared. $threshold$ is a lower bound threshold value. If the upper bound for c_i exceeds the threshold or, in other words, whenever there is a chance that this document may achieve a better similarity score than the $threshold$, the document is inserted according to its upper bound similarity into the priority queue. Then we compute the similarity for the document, that we can guarantee it will achieve (lower bound). If this more than the current $threshold$, we can be more selective for the following documents and therefore updated $threshold$ to this value. This way, if one document's upper bound similarity does not exceed $threshold$, we know that this document can even in the best case not achieve a better similarity than some document, which we already added to our candidate set. Hence, we do not need to pay further attention to it and, especially, do not need to perform the expensive Hungarian algorithm on it.

The second stage takes the document with the highest upper bound similarity. If we do not know the exact document similarity yet, it is now computed using the Hungarian algorithm. The result is stored again in the priority queue. By this, we can guarantee that whenever we pick a document from the queue for which we already computed the exact similarity, it has to be the most similar document in the entire collection since none of the remaining candidates has higher upper bound similarity.

This algorithm can easily be extended to return an ordered list of documents (Top-k results) instead of just one. The only two changes needed are (1) make sure that threshold in line 9 is set to the k-th highest lower bound and (2) continue to execute the second phase until k documents have been returned (or the heap is empty, because C did not contain k documents).

3 Related Work

Much effort has been spent on finding appropriate algorithms to match objects of identical type in various fields.

For instance, information retrieval developed several models for finding text in large document collections, starting from Boolean retrieval, reaching to vector space and probabilistic models, which can be used for matching complete text documents. Inexact matches in text and coded information can be found using a thesaurus. Since there exist many different coding standards for medical information, in 1986 the National Library of Medicine (NLM) started a project to build a meta-thesaurus named Unified Medical Language System (UMLS) [10]. The improvement of retrieval effectiveness exploiting a controlled vocabulary in medicine have been studied e.g. in [11].

Content-based image retrieval frequently employs numeric features like histograms, wavelet coefficients, Fourier descriptors, etc. to express low level image characteristics like color distribution, texture and shape properties. An overview on existing techniques for content-based image retrieval with respect to applications in medical imaging is presented in [1]. Despite of those *low-level* features, the so-called *semantic gap* between a user's perception of alike images and computed image similarity persists. Research has come up with complex, object-centered query models that incorporate and weight various features of different query objects.

Algorithms for similarity search in time series caught attention within the last decade (e.g. [12]) in the database and data mining community. But as [13] remarks, many of these are not applicable to any domain. Similarity search in the biomedical domain is widely used in sequence alignment, for which the most prominent example is BLAST [14].

Compared to similarity on individual object domains, complex multimedia documents that comprise many different media objects have received little attention, so far. The main focus has been on compound documents containing images and texts. E.g. [15] provides a model for querying documents according to their structure, text, and image components. The authors of [16] propose a query language for structured multimedia documents named POQL$^{\text{MM}}$, which is capable of including audio and video content and uses conversation functions to combine this with text retrieval. The language is described in more detail in [17]. However, in both approaches it is not possible to prohibit duplicate assignment of objects to several query objects. Many systems, especially in the medical domain, like CBIR2 [18] for instance, have the structural restriction that each document can contain only one text object and exactly one image. For this reason, their usage for general digital libraries of compound documents is very limited.

Other approaches generate textual annotations, which can be queried based on keywords. For this purpose, CIMWOS [19] uses audio, text, and image processing and [20] uses object recognition and spatial orientation. These approaches cannot provide *query-by-example* based on similarity measures requiring non-textual represented features, like nearest neighbor image search base on shape, which would be needed for some applications. Even the opposite approach is thinkable: [21] describes a system to characterize documents only based on visual similarity, not employing any textual data. This approach is clearly restricted to an even smaller application domain.

4 Summary

We proposed a novel approach to compound document matching in which it is possible to use existing similarity functions for component objects. Specifically, we have adapted a *filter-and-refinement* algorithm from region-based image retrieval to cope with fundamental concepts of similarity search in medical information systems. That is, we have introduced the notion of *domains* to incorporate the inherently heterogeneous structure of complex patient records. A domain subsumes distinct characteristics of a component object including its media type (e.g. text, image, signal, numeric, etc.), the origin (e.g. CT, MRI, sonography), and the subject (e.g. organ, tissue, etc.).

In terms of query processing, we exploit this concept to exclude cross-domain matchings of component objects. As computing the similarity matrix is the most time-consuming part of document matching, we achieve noticeable speed-ups as opposed to the native algorithm.

5 Outlook

For the time being, this paper has classified patient records as instances of plain multimedia documents. Precisely, matching rests upon equally weighted incoherent component objects. Future extensions of our matching approach must incorporate a complex query model that allows the adjustment of influence of component objects by assigning individual weights, but also covers temporally and structurally interrelated content, like treatments conducted on a diagnosis given at a certain point in time. A clear separation of self-contained component objects and metadata that attributes these objects will (1) address effectiveness of object matching incorporating additional object-centric attributes and (2) increase efficiency of document matching by having fewer component objects. The query model also needs to be extended to deal with predicate-like exact matching conditions.

There is still some room left for improvement of the retrieval efficiency of the two-stage filter-and-refinement algorithm. The algorithms can easily be parallelized, e.g.

distributing the computationally intensive task of solving the assignment problem to a Grid infrastructure. The next intensive task of computing the individual domain similarity matrices is also a candidate for distributed computing. Another, orthogonal approach would be to incrementally narrow the upper and lower document similarity by computing the exact similarity first only on the domains, which have the highest possible impact of changing to the document similarity. By this, a (near) optimal multi-step approach similar to [22] can be achieved. Performance evaluations on synthetic and real-world data have to proof the gained speed-up.

In general, two different patient records will seldomly contain the same number of objects in the same domains. Query processing must cope with incomplete patient records that lack certain component objects. Penalties, as introduced by RBIR algorithms, do not fully reflect this scenario. Therefore, further investigation of the choice of penalty values and assigning weights to specific domains is necessary. Cross-domain object matching based on domain knowledge might relieve the situation of non-complementary object domains, e.g. a textual diagnosis containing the term "myocardial infarction" can be matched with a cardiogram that exhibits typical patterns.

References

1. Müller, H., Michoux, N., Bandon, D., Geissbuhler, A.: A review of content–based image retrieval systems in medical applications — clinical benefits and future directions. International Journal of Medical Informatics **73** (2004) 1–23
2. Vogl, R., Berreck, M., Pellizzari, T., Pirchl, C., Reiter, D., Schwab, M., Stark, C., Wallinger, M., Wilhelmy, I.: The innsbruck advanced image management (aim) project: a comprehensive archive for large volume medical data with tight integration to the cis/epr. In: gmds2004: 49. Jahrestagung der Deutschen Gesellschaft fr Medizinische Informatik, Biometrie und Epidemiologie (gmds), gms (2004)
3. Weber, R., Mlivoncic, M.: Efficient region-based image retrieval. In: CIKM '03: Proceedings of the twelfth International Conference on Information and Knowledge Management, ACM Press (2003) 69–76
4. Knuth, D.E. In: The StanfordGraph Base. ACM Press (1993) 41–43
5. Weber, R., Schek, H.J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, Morgan Kaufmann (1998) 194–205
6. Böhm, K., Mlivoncic, M., Schek, H.J., Weber, R.: Fast evaluation techniques for complex similarity queries. In: VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc. (2001) 211–220
7. Bartolini, I., Ciaccia, P., Patella, M.: A sound algorithm for region-based image retrieval using an index. In: DEXA '00: Proceedings of the 11th International Workshop on Database and Expert Systems Applications, IEEE Computer Society (2000) 930–934
8. Maintz, J.B.A., van der Elsen, P.A., Viergewer, M.A.: Comparison of feature-based matching of ct and mr brain images. In: CVRMed '95: Proceedings of the First International Conference on Computer Vision, Virtual Reality and Robotics in Medicine, Springer-Verlag (1995) 219–228
9. Lehmann, T.M., Schubert, H., Keysers, D., Kohnen, M., Wein, B.B.: The irma code for unique classification of medical images. In: Proceedings of SPIE Medical Imaging 2003: PACS and Integrated Medical Information Systems. Volume 5033. (2003) 440–451

10. Lindberg, D.A., Humphreys, B., McCray, A.: The unified medical language system. Methods of Information in Medicine **32** (1993) 281–291
11. French, J.C., Powell, A.L., Gey, F., Perelman, N.: Exploiting a controlled vocabulary to improve collection selection and retrieval effectiveness. In: CIKM '01: Proceedings of the tenth international conference on Information and knowledge management, ACM Press (2001) 199–206
12. Agrawal, R., Faloutsos, C., Swami, A.N.: Efficient similarity search in sequence databases. In: FODO '93: Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms, Springer-Verlag (1993) 69–84
13. Keogh, E., Kasetty, S.: On the need for time series data mining benchmarks: a survey and empirical demonstration. In: KDD '02: Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press (2002) 102–111
14. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.: Basic local alignment search tool. Journal of Molecular Biology **215** (1990) 403–410
15. Meghini, C., Sebastiani, F., Straccia, U.: Modelling the retrieval of structured documents containing texts and images. In: ECDL '97: Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries, Springer-Verlag (1997) 325–344
16. Henrich, A., Robbert, G.: Combining multimedia retrieval and text retrieval to search structured documents in digital libraries. In: Pre-Proceedings of the 1st DELOS Workshop on Information Seeking, Searching and Querying in Digital Libraries. (2000)
17. Henrich, A., Robbert, G.: Poqlmm: A query language for structured multimedia documents. In: Proceedings of the 1st International Workshop on Multimedia Data and Document Engineering (MDDE01). (2001) 17–26
18. Antani, S., Long, L.R., Thoma, G.R.: A biomedical information system for combined content-based retrieval of spine x-ray images and associated text information. In: Proceedings of the Indian Conference on Computer Vision, Graphics, and Image Processing (ICVGIP 2002). (2002)
19. Papageorgiou, H., Protopapas, A.: Cimwos: A multimedia, multimodal and multilingual indexing and retrieval system. In: Proceedings of the 4^{th} European Workshop on Image Analysis for Multimedia Interactive Services, World Scientific Publishing Co. (2003) 563–568
20. Rabitti, F., Savino, P.: An information-retrieval approach for image databases. In: Proceedings of the 18th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc. (1992) 574–584
21. Bagdanov, A.D., Worring, M.: Granulometric analysis of document images. In: 16 th International Conference on Pattern Recognition (ICPR'02) Volume 1. (2002)
22. Seidl, T., Kriegel, H.P.: Optimal multi-step k-nearest neighbor search. In: SIGMOD 1998: Proceedings ACM SIGMOD International Conference on Management of Data, ACM Press (1998) 154–165

User-Adaptable Browsing and Relevance Feedback in Image Databases

Dragana Damjanovic, Claudia Plant, Sören Balko, and Hans-Jörg Schek

University for Health Sciences, Medical Informatics and Technology
Eduard-Wallnöfer-Zentrum 1 A–6060 Hall in Tirol Austria
[dragana.damjanovic|claudia.plant|sören.balko|hans-joerg.schek@umit.at]

Abstract. In this paper, we present the outline of a new interaction technique for searching and browsing in large image collections. The main idea is to learn an adaptive similarity measure from user interaction that helps to overcome the so-called semantic gap. We give an overview on the contents of the database by clustering the images and displaying representatives of these clusters. We use relevance feedback information from the user to refine our clustering by adapting the similarity measure.

1 Introduction

Images represent a huge part of the information stored in digital libraries. Content-based image retrieval (CBIR) has become a popular paradigm for similarity search on images that lack textual annotations. The major challenge is the so-called *semantic gap* between machine-computed similarity and human perception of alike images. Multimedia information retrieval employs numeric features like histograms, wavelet coefficients, Fourier descriptors etc. that express plain image characteristics like color, texture, and shape. Those features are assembled to high-dimensional feature vectors, represented as points in vector space. In this way, similarity search corresponds to nearest neighbor or range queries on top of those low-level features and a given distance. Users, however, identify similar images on a semantically higher level, like images portraying same persons. Numerous approaches have come up with relevance feedback techniques to bridge the semantic gap. Being an integral part of query processing, user interactions, like result evaluations enhance effectiveness of similarity search. As image databases grow rapidly, average users find it increasingly difficult to express their information needs as complex predicates. This paper introduces an interactive browsing and relevance feedback approach that rests upon interactive clustering. In this way, we go beyond the traditional query-by-example paradigm in CBIR. We start by delivering the rough content of the image database by visualizing its cluster structure. Especially for large collections an initial k-nearest neighbors query with a large number of k can also be a good starting point. Users can narrow down the result by branching into clusters that contain relevant images while discarding irrelevant ones. Additionally, users provide relevance feedback in order to modify the cluster structure. This interactive process transforms feature space to a user's information needs.

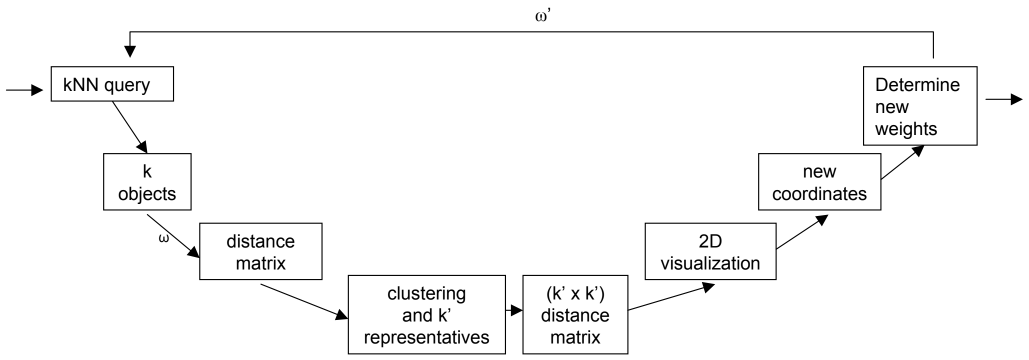


Fig. 1. Relevance Feedback by Interactive Clustering - Steps.

The remainder of this paper is organized as follows. The next section provides a formal problem specification. Section 3 introduces our user-adaptable browsing approach and we report on first experimental results. Section 5 briefly discusses related work on relevance feedback, browsing techniques in CBIR, and clustering. Section 6 concludes this paper.

2 Problem Specification

Given a DB of N data objects and let be $k \in N$ a subset of these objects, e. g. obtained by an initial k-nearest neighbor search. We introduce the notions *feature distance* and *combined distance* to formalize the similarity among them. Different types of features can be used to represent an image, e. g. color or texture. Let $F = \{F_1, ...F_m\}$ be the set of feature types. For each feature type F_i we have N feature vectors of dimensionality d_i. The feature distance $Dist_{F_i}$ of the objects P and Q is their distance regarding the feature type F_i. Let $\pi(P)$ be the projection on the i-th feature type of P. As distance function, denoted by $Dist_{F_i}(P,Q)$ we use a weighted metric distance function, e. g. Euclidean Distance.

$$Dist_{F_i}(P,Q) = \sqrt{\sum_{j=1}^{d_i} \omega_{i,j} \cdot |\pi_i(P) - \pi_i(Q)|^2}. \quad (1)$$

To express the similarity of objects P and Q taking all feature types into account, we compute the combined distance $Dist_c$, which is a weighted sum of the feature distances.

$$Dist_c(P,Q) = \sum_{k=1}^{m} \omega_i \cdot Dist_{F_k}(P,Q). \quad (2)$$

From the information obtained by relevance feedback, we want to adjust the weights $\omega = \omega_i \cup \omega_{i,j}$ respecting the following condition.

$$\lambda \cdot \sum (\delta_\omega)^2 + (1-\lambda) \sum v_{x,y}^2 \rightarrow min. \qquad (3)$$

Figure 1 depicts the steps in our method for relevance feedback by interactive clustering. Starting from the distance matrix with the n objects computed using initial weights ((1), (2)) we perform a clustering step and chose n' representatives. We display these objects in a 2-dimensional user-interface. To obtain approximate 2D-coordinates, we use a distance preserving projection like FastMap [1]. The user now can interactively change the cluster structure by moving objects. We use this information as relevance feedback to determine new weights by solving (3). (3) means that we minimize v_x, v_y, i. e. the deviation from the coordinates a user generated by moving by suitable weights. If a user does not move any point that means that no weight changes will result. We want to keep the change of weights δ_ω small. In the general case by solving the generalized least square problem (3) there will be always a unique set of new weights that minimizes (3). The factor λ can be the smaller the more movements has been performed.

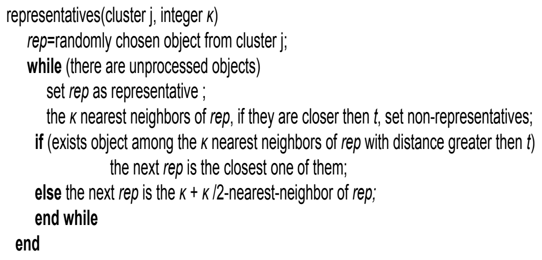
3 User-adaptable Browsing by Interactive Clustering

In contrast to other approaches we use a clustering algorithm to provide the user a better overview on the displayed objects. The user then can interactively change the cluster structure. In addition, clustering allows to change the distance function locally, i. e. we use different similarity measures between the clusters. In this section, we describe one possible solution to realize the single steps depicted in figure 1.

3.1 Clustering

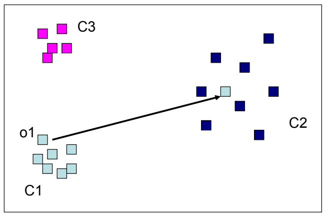
To discover the hidden data structure of the collection or the result set of an initial k-nearest neighbors query, we first apply a clustering algorithm. For higher dimensional feature vectors, e. g. features derived from different regions of images, there probably has to be a dimensionality reduction step before clustering. We compared different clustering algorithms, in particular k-Means [2] and DBSCAN [3]. The partitioning clustering algorithm k-Means starts by arbitrarily partitioning the data set in k disjoint set of objects and computing their mean vector, the so-called cluster center. In an iterative optimization process each object is then assigned to its closest cluster center and the cluster centers are updated. This step is repeated until no changes of the cluster centers occur from one iteration to the next. K-Means has no determinate result and the number of clusters has to be specified in advance. The algorithm always finds k clusters that may not properly correspond to the used similarity measure.

Density-based clustering methods regard clusters as connected dense regions in the feature space that are separated by regions of lower density. Therefore density-based clusters give a good impression of the currently used similarity



```
representatives(cluster j, integer κ)
    rep=randomly chosen object from cluster j;
    while (there are unprocessed objects)
        set rep as representative ;
        the κ nearest neighbors of rep, if they are closer then t, set non-representatives;
      if (exists object among the κ nearest neighbors of rep with distance greater then t)
                the next rep is the closest one of them;
      else the next rep is the κ + κ /2-nearest-neighbor of rep;
      end while
  end
```

(a) Selecting Representatives.



(b) Interactive Clustering.

Fig. 2.

measure. In the algorithm DBSCAN [3] this idea is formalized using two parameters, $MinPts$ specifying a minimum number of objects, and ϵ, the radius of a hyper-sphere. These two parameters determine a density threshold for clustering. An object is called a *core object* of an $(\epsilon, MinPts)$-cluster if there are at least $MinPts$ objects in the ϵ-neighborhood. If one object P is in the ϵ-neighborhood of a core-object Q, then P is said to be *directly density-reachable* from Q. The *density-connectivity* is the symmetric, transitive closure of the *direct density reachability*, and a *density-based* $(\epsilon, MinPts)$*-cluster* is defined as a maximal set of density-connected objects. DBSCAN has a determinate result, finds arbitrary shaped clusters and the number of clusters does not has to be specified in advance.

3.2 Selecting Representatives and Visualization

In contrast to partitioning clustering algorithms like k-Means, there are no cluster centers or central objects using DBSCAN. Since we usually cannot display all the clustered objects in our interface, we have to select some representatives of the clusters which are displayed. Pseudo-code of algorithm for choosing representatives for each cluster is given in figure 2(a).

3.3 Interactive Clustering

Locally Determining Weights. In our approach clustering is done for two reasons. Firstly, to give the user an impression of the currently used similarity measure. Since clustering groups similar objects together, the user can evaluate if the currently used weights correspond to his notion of similarity. A common problem with displaying high dimensional data like images in a 2D-space is the unavoidable information loss due to the radical dimensionality reduction. The arrangement of objects in the display interface can only give an approximate impression of the similarity in the full dimensional space. Since we display the images in differently colored frames indicating their cluster assignment, the user can better see the true underlying similarity measure.

But even more important, we apply clustering to be able to change the weights used for the distance function locally. Assume we have 3 clusters of images displayed (cf. figure 2(b)). If the user only moves objects from cluster $C1$ to cluster $C2$ and cluster $C3$ remains unchanged, we also should internally change only the similarity measure between objects of cluster $C1$ and $C2$. We implement this idea by locally differently weighting the similarity measure. We use the input of the user to learn different similarity measures that are locally applied in different areas of the feature space. We store the weights in a *local similarity matrix* S. S is a $|k| \times |k|$ matrix, with $S(i, j) = \omega(i, j)$ for $i,\ j \in [1..|k|]$. This means $S(i, j)$ contains the currently used weight $\omega(i, j)$ for the similarity measure between objects of cluster i and cluster j. Before having any user input, we used un-weighted distances, i. e. we set $\omega(i, j) = 1\ \forall i, j$. In the next section, we describe how to change the weights using relevance-feedback.

Changing the Clustering by Relevance-Feedback. The user can change the cluster structure of the displayed objects according to his notion of similarity. A simple example is depicted in figure 2(b). The user moves the object o_1 from cluster $C1$ to cluster $C2$, since in his opinion the object rather belongs to this cluster. In this case, we have to find out

1. why o_1 is similar to the objects in cluster $C2$,
2. if there are any objects besides o_1 in $C1$ which should be moved, too.

While the first aspect is obvious, the second aspect may not be that clear. After changing the local similarity matrix due to user interaction, we may move more objects than the ones the user explicitly has moved. Some objects or all of the objects of the affected clusters may have to be moved according to the new similarity measure. We partially implemented this idea by internally transforming the feature space using Linear Discriminant Analysis [4]. We use LDA for both (1.) assigning objects to other clusters and (2.) splitting up clusters. LDA is a well-known feature extraction and dimensionality reduction technique that has been successfully used in many pattern recognition problems. Given data objects o_i and a set of groups or classes c_i (in our case the objects belonging to the clusters $C1$ and $C2$) LDA finds the linear transformation t in which the c_1 and c_2 are separable in the best way. t is the transformation that maximizes between-class separability while minimizing within-class variability, formally:

$$\text{argmax } \tilde{S}_B := \sum_{i=1}^{c} (\tilde{\mu}_i - \tilde{\mu})(\tilde{\mu}_i - \tilde{\mu})^T$$

$$\min \tilde{S}_W := \sum_{i=1}^{c} (x - \tilde{\mu}_i)(x - \tilde{\mu}_i)^T$$

where $\tilde{\mu}_i$ is the mean of the projected data objects of class c_i, $\tilde{\mu}$ is the mean of all projected data and objects and c is the number of classes. $\tilde{S}_B$ is called

(a) FastMap. (b) IsoMap.

Fig. 3. Cluster Representatives.

the between-scatter matrix, $\tilde{S}_W$ the within scatter matrix, respectively. That corresponds to finding the matrix of column feature vectors V that maximizes $V^T S_B V$ and minimizes $V^T S_W V$ where S_W and S_B are the untransformed within and between scatter matrixes. Thus, the following objective function has to be maximized.

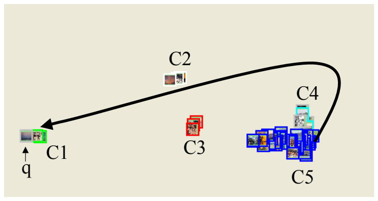
$$J(V) = \frac{|V^T S_B V|}{|V^T S_W V|}.$$

It can be shown [4] that finding V such that $J(V)$ is maximized corresponds to solving the eigenvector problem $S_B V = \lambda S_V V$. Since the rank of S_B is at most $c-1$, at most $c-1$ features can be extracted. For a two-class-problem, this means we get the most discriminant direction $\overrightarrow{u}$. We directly use $\overrightarrow{u}$ as new weights when splitting two clusters. Let us note that $\overrightarrow{v} \perp \overrightarrow{u}$ is the direction in which two groups of objects are the most similar. We exploit this fact when assigning objects to other clusters.

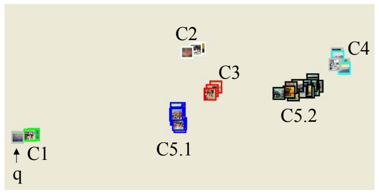
LDA and the Least Squares Method. In section 2 we gave a formal problem specification. We want to determine new weights using relevance feedback while minimizing equation (3). W.l.o.g. we restrict ourselves to the case of 2 displayed clusters (this means globally changing weights). We can not prove property (3) since we use non-linear or heuristic mappings to 2D space such as IsoMap [5] and FastMap [1], and found no way to reverse them till now. We also tried to use PCA, but the visualization result is not satisfactory. But nevertheless there is a strong connection between LDA and the least square method. When we use LDA to split or merge clusters, our goal is to find the projection of the feature space w. r. t. which the objects of the clusters affected by user interaction are the most different or the most similar, respectively. This can be done with a linear least squares method, too. One is looking for a discriminant function between two groups of objects including a bias term b, i. e.

$$f(x) = v^t x + b$$

such that the sum of squared errors between $f(x)$ and the known true class assignment of each object o_i, (denoted by $c(o_i)$) is minimized. Often, $c(o_i)$ corresponds to the cluster assignment of o_i, but i. g. when splitting clusters, we



(a) Starting.



(b) After movement.

Fig. 4. Query-by-Example.

consider various subgroups of clusters, therefore we a using the notion *class* here. It can be shown [6], that minimizing the sum of squared errors

$$E(w,b) = \sum_{o_i} (f(x) - c(o_i))^2 = \sum_{o_i} (v^T x + b - c(o_i))^2$$

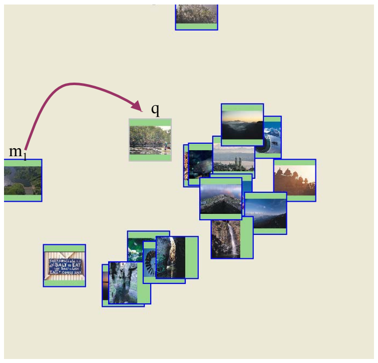
leads to the same projection t as found by LDA. We are also considering alternative ways to implement splitting and merging of clusters and computing the weights, e. g. non-linear least square methods or the use of kernel-functions. In the next section, we explain in more detail how the weights are determined showing example screenshots.

4 First Experiments

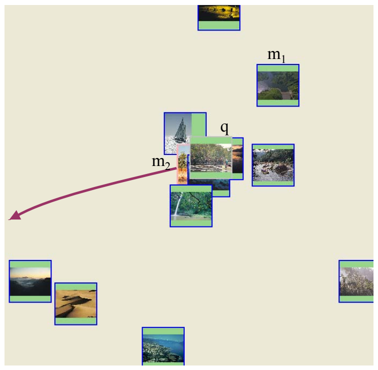
For our first experiments we use an example collection of images from different sources, as used in [7]. In particular we have 9-dimensional feature vectors consisting of color moments in the Lab-color space [8]. In this case $F = \{F_1\}, d_1 = 9$ and we want to adjust the weights ω_j of the feature distance $Dist_{F_1}$. We have implemented a simple 2-dimensional user-interface. In order to compare different mapping algorithms, we implemented FastMap [1] and IsoMap [5]. We also implemented IsoMap with underlying FastMap instead of MDS, since this makes computation faster. In our prototype the user can choose between these different mappings. In figure 3 screenshots are depicted showing the representatives of different clusters as thumbnails. Currently we are performing first tests on interactive clustering. Besides our example collection we use synthetic data to have more control on the results.

4.1 Query-by-Example

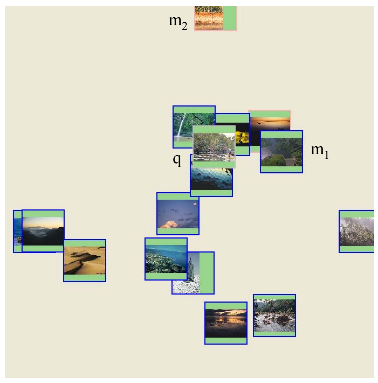
Starting with a query image, we perform a k-nearest neighbors query with Euclidian distance and initial weights $\omega_j = 1$. Initially we extract a large number of k-nearest neighbors (around 500). Our Interface gives the user opportunity to browse trough this collection of nearest neighbors of the query image giving an overview of them. This is internally implemented by clustering query results



(a) Starting.



(b) First movement.



(c) Second movement.

Fig. 5. Query-by-Example.

using DBSCAN and Euclidian distance with initial weights and showing representatives of the clusters (cf. algorithm in figure 2(a)).In figure 4(a) an example is depicted showing the query image (denoted by q) and 5 clusters. Assume the user found in cluster $C5$ an image that are similar to the query image and moves it close to it. We than split the cluster. To implement that, we want to firstly find out the difference between the moved image and other images in the cluster and secondly, why the moved image is similar to the query image.
We apply two times LDA to get this information. We threat the query image q and its k-nearest neighbors like an additional cluster C_q to be able to properly determine weights between the region surrounding q and the other clusters. The local similarity matrix S is a $|(k+1)| \times |(k+1)|$ matrix in the case of Query-by-Example. In the first step, we find the most discriminant projection t between the moved image m and its k-nearest neighbors (they form one subgroup c_1) and the object o in the same cluster which is the furthest away from m . o and its k-nearest neighbors form the subgroup c_2. We weight the distance function between c_1 and c_2 using t. We now assign each object of the cluster either to c_1 or

c_2, by computing the distance to m and o and assigning it to the closer object. In our example, the cluster $C5$ is split up in two parts: the objects of group c_1 form the cluster $C5.1$ and the objects of group c_2 $C5.2$, respectively. In a second step, we use LDA to find out why $C.5.1$ and C_q are similar. Analogously we compute an orthogonal hyperplane to the most discriminant projection t between the C_q and the group c_1. As result $C5.1$ is closely displayed to the query image after applying FastMap again (cf. figure 4(b)). Now the user could select one cluster, e. g. the most relevant one to the query, and re-run the query using the weights between C_q and that cluster.

Our second example depicted in figure 5 shows a later stage of the interaction process. The user has already focussed on one cluster. Results of one query with initial weights and query image q are shown on picture 5(a). For example user finds image m_1 similar to query image q and moves it closer. As described above, we determine new weights between different subgroups of the cluster, corresponding to the region around q and m_1. In case of approaching an image to the query image, we use an orthogonal direction to the discriminant projection t to weight the disstance function. The results after a new k-nearest neighbors query with $k = 30$ are are shown in 5(b). Note that we have a different result set now. In the next step user move image m_2 away from the query image (user find it irrelevant for his query). Here, we analogously use the most discriminant projection t between the two subgroups of the cluster. The final results are shown in figure 5(c).

5 Related Work

In this section, we discuss related work on CBIR, relevance feedback and browsing. Besides that, we give a short introduction on feature extraction and dimensionality reduction techniques.

5.1 Feature Extraction and Dimensionality Reduction

The complex meaning of image is represented as a feature vector. Feature vectors are extracted from the color, shape, texture and regions of the image content. For extracting the feature vectors of images there are different techniques, among them the most popular are color histograms, like RGB histograms, HSV color histograms, HMMD color histograms and other techniques.
Representing complex content of images as feature vectors is just one problem that arise in area of image retrieval. This problem is called *semantic gap*. The next problem following is the problem of presenting high dimensional feature space in low dimensional space, like 3 dimensional space, the maximal number of dimensions human vision is able to capture. A mapping algorithm from feature space to the visualization space is needed. The classical techniques for dimensionality reduction, PCA and MDS, are simple to implement and guarantee to discover the true structure of data lying on or near a linear subspace of the

high dimensional feature space. Among them is also FastMap [9] which is more efficiently computable. There are some recent methods like ISOMAP (ISOmetric MAPping)[5], SNE (Stochastic Neighbor Embedding)[10], LLE (Local Linear Embedding) [11]. These methods are nonlinear and they are trying to preserve the inner geometrical structure of the data. However an exact match between feature space and visualization space is impossible.

5.2 CBIR and browsing

Many CBIR(Content-based Image Retrieval) systems are based on the concept *Query-by-example*[9] [12] In order to improve the retrieval, CBIR systems often employ relevance feedback, in which the user can refine the search incrementally by giving feedback to the result of the previous query. In many cases relevance feedback is based on labeling a set of retrieved images according to their importance and it is usually perform in one of two ways: query point moving or weight updating.
Query by Groups[13] is an extension to *Query by Example* mode described above. The difference is that *Query by Example* handles image individually, in *Query by Group*, a group of images are considered as a unit of the query. Relevance feedback is establish in the same way by marking, in this case, group of images as positive or negative examples. In the beginning the system displays randomly selected images, which are not reflection of the real content of data set.
In [14] the authors focus on browsing the collection, more then on querying by image or group of images. The key idea in NN^k network is to store the oriented graph of images that were retrieved as similar to that image under some feature regime.In order to allow searches without any query, they provide the user with a representative set of images from the collection by clustering nodes up to a certain depth.The use browses trough collection by chousing image, and as the results a network of its nearest neighbor is shown. In this paper no relevance feedback is proposed.
There are also built some interfaces which are combining browsing and querying with possibility for user to give relevance feedback, e. g. [15]. At the beginning it is given information about the status of whole database and the user manipulate the image space directly by moving images around. The manipulation of images in the display cause the creation of a similarity measure that satisfies the relations impose by user. In contrast to that approach, we use clustering and we change similarity measure locally.
In broader sense, (semi-)supervised clustering is related to our approach, e. g. [16]. been shown that classified examples can improve the results of partitioning clustering algorithms. In [17] a semi-supersived version of the EM-Algorithm is proposed for image retrieval. This approach suffers from the drawbacks that the number of clusters has to be specified in advance and there is no transformation of the feature space to better fit the users needs.

6 Conclusion

In this paper, we presented first ideas on browsing and visual relevance feedback in image collections by interactive clustering. Since image collections are rapidly growing, our approach is focussing particularly on k-nearest-neighbor queries with a large number of k. Our interface offers a clearly arranged presentation of the huge result set of such queries using clustering and properly selecting cluster representatives which are displayed in 2D-space. The user then browses through the collection and interactively changes the cluster structure by moving objects. Using this information as relevance feedback we gradually adjust the similarity measure according to the users needs. We find subspaces of the feature space in which the objects are similar or dissimilar, respectively. We locally change the similarity measure, i. e. we only take objects in the areas affected of the users movements into account. Due to this local changes the user has a better control on the effects of relevance feedback, especially in large collections. There are still many open issues, e. g. in order to capture the input given by moving the images more accurately, we need an inverse of the mapping from the 2-dimensional visualization space back to the high dimensional feature space.
Other directions we want to elaborate are e. g. creating and managing different user profiles or to displaying different feature types to 2D spaces. Another interesting option would be to display the objects in different 2D-spaces using different similarity measures.

References

1. Faloutsos, C., Lin, K.I.: "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets". In: SIGMOD Conference. (1995) 163–174
2. MacQueen, J.: "Some Methods for Classification and Analysis of Multivariate Observations". In: 5th Berkeley Symp. Math. Statist. Prob. (1967)
3. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96). (1996)
4. Fisher, R.A.: "The Use of Multiple Measurements in Taxonomic Problems". In: Annals of Eugenics. (1936)
5. Tenenbaum, J.B., de Silva, V., Langford, J.C.: "A Global Geometric Framework for Nonlinear Dimensionality Reduction". In: Science 290(5500). (2000) 2319–2323
6. Park, C.H., Park, H.: "A Relationship between LDA and the Generalized Minimum Squared Error Solution". In: SIAM Journal on Matrix Analysis and Applications. (2004)
7. Böhm, K., Mlivoncic, M., Schek, H.J., Weber, R.: "Fast Evaluation Techniques for Complex Similarity Queries". In: VLDB. (2001) 211–220
8. Stricker, A., Dimai, M.: "Spectral Covariance and Fuzzy Regions for Image Indexing". In: Machine Vision and Application. (1997) 10:66–73
9. Ishikawa, Y., Subramanya, R., Faloutsos, C.: Mindreader: Querying databases through multiple examples. In: VLDB. (1998) 218–227
10. Hinton, G.E., Roweis, S.T.: Stochastic neighbor embedding. In: NIPS. (2002) 833–840

11. S.Roweis, Saul, L.: Nonlinear dimensionality reduction by locally linear embedding. In: Science 290(5500). (2000) 2323–2326
12. Rui, Y., Huang, T.S.: Optimizing learning in image retrieval. In: CVPR. (2000) 1236–
13. Nakazato, M., Manola, L., Huang, T.S.: Imagegrouper: a group-oriented user interface for content-based image retrieval and digital image arrangement. J. Vis. Lang. Comput. **14** (2003) 363–386
14. Heesch, D., Rüger, S.M.: "NN^{k} Networks for Content-Based Image Retrieval". In: ECIR. (2004) 253–266
15. Santini, S., Gupta, A., Jain, R.: "Emergent Semantics through Interaction in Image Databases". IEEE Trans. Knowl. Data Eng. **13** (2001) 337–351
16. Eick, C., Zeidat, N., Zhao, Z.: " Supervised Clustering - Algorithms and Benefits". In: Proc. of the International Conference on Tools with AI ICTAI. (2004)
17. Dong, A., Bhanu, B.: "A New Semi-Supervised EM Algorithm for Image Retrieval". In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003). (2003) 662–667

Indexing the Distance Using Chord: A Distributed Similarity Search Structure

David Novák and Pavel Zezula

Faculty of Informatics, Masaryk University, Brno, Czech Republic
{xnovak8,zezula}@fi.muni.cz

Abstract. The need of search mechanisms based on data content rather then attributes values has recently lead to formation of the metric-based similarity retrieval. The computational complexity of such retrieval and the large volume of processed data call for distributed processing. In this paper, we propose *chiDistance*, a distributed data structure for similarity search in metric spaces. The structure is based on the idea of a vector-based index method *iDistance* which enables to transform the issue of similarity search into the one-dimensional range search problem. A Peer-to-Peer system based on the *Chord* protocol is created to distribute the storage space and to parallelize the execution of similarity queries. In the experiments conducted on our prototype implementation we study the system performance concentrating on several aspects of parallelism of the range search algorithm.

1 Introduction

In traditional data retrieval, the query specifies a pattern to exactly match attributes of the required data. The present-day systems that manage complex data types (such as images, videos, text documents or DNA sequences) require search mechanisms based on the data content rather then data attributes. Therefore, the field of *content-based* or *similarity* retrieval has made a rapid progress recently. In principle, a similarity query works as follows: given a query data object q, the search process extracts all indexed data objects that are "similar" to q. The similarity of data can be generally defined by a dissimilarity function (*distance function*) d that is measurable for every pair of objects. The data set together with the distance function can be seen as a metric space.

Many metric index structures have been proposed – see recent surveys [1, 2]. In real-life applications, the distance function d is typically expensive to compute. This fact, together with the volume of the data being managed nowadays, lead to the need of distributed processing. Most of the recent effort in the field of distributed indexes for similarity search has concentrated on the vector (attribute) data (see, e.g., [3–5], or SWAM [6]). As far as we know, GHT* index, proposed in [7], is the only published metric-based distributed data structure.

Recently, network architecture paradigms referred to as Peer-to-Peer (P2P) and Grid systems [8] have been gaining in popularity. In short, P2P structures are distributed systems without any hierarchical organization where each node is running software with equivalent functionality. This concept is attractive for our work because of its scalability and self-organizing nature.

In this paper, we introduce a new distributed data structure called *chiDistance*. It is based on the idea of *Indexing the Distance* (*iDistance*) [9] and generalizes this attribute-based index method to become a metric-based method. Through this concept, the issue of similarity search is transformed into the one-dimensional range search problem. Then, the P2P protocol *Chord* [10] is used to form the distributed structure for *chiDistance*. The *iDistance* search algorithms are generalized to the proposed architecture.

The paper is organized as follows. Section 2 provides background for the metric-based similarity search and describes two techniques that are essential for this paper – *iDistance* and *Chord*. In Section 3, we describe the proposed *chiDistance* data structure in details. Section 4 presents results of the performance experiments and the paper concludes in Section 5 with directions for our future work.

2 Preliminaries

In the scope of our work there are general similarity search structures that are not limited to any specific data set or application. *Metric space* is a suitable structure to model data set and relationships between data objects.

2.1 Metric Space Searching

Mathematically, metric space $\mathcal{M}$ is a pair $\mathcal{M} = (\mathbb{U}, d)$, where $\mathbb{U}$ is the *domain* of objects and d is the total *distance function* $d : \mathbb{U} \times \mathbb{U} \longrightarrow \mathbb{R}$ satisfying the following conditions for all objects $x, y, z \in \mathbb{U}$:

$$\begin{array}{ll} d(x,y) \geq 0 \quad \text{and} \quad d(x,y) = 0 \text{ iff } x = y & (\textit{non-negativity}), \\ d(x,y) = d(y,x) & (\textit{symmetry}), \\ d(x,z) \leq d(x,y) + d(y,z) & (\textit{triangle inequality}). \end{array}$$

Let us define two essential types of similarity queries: the *range query* and the *k nearest neighbors query* [2]. Let $I \subseteq U$ be a finite set of indexed objects.

Definition 1. *Given an object* $q \in \mathbb{U}$ *and a* maximum search distance r*, the* range query $\mathbf{Range}(q, r)$ *selects all objects* $x \in I$ *such that* $d(q, x) \leq r$.

Definition 2. *Given an object* $q \in \mathbb{U}$ *and an integer* $k \geq 1$*, the* k nearest neighbor query $\mathbf{kNN}(q, k)$ *selects the* k *objects* $x \in I$ *which have the shortest distances from* q.

The presented data structures focus on these types of similarity queries.

2.2 Indexing the Distance

The *iDistance* [9] is an indexing method for similarity search in vector spaces. It partitions the data space into clusters and selects a reference object (pivot) p_i for each cluster C_i, $0 \leq i \leq k$. Every data object is assigned a one-dimensional *iDistance* key according to the distance to its cluster pivot. Having a cluster separation constant c, the *iDistance* value for an object $x \in C_i$ is

$$iDist(x) = d(p_i, x) + i \cdot c.$$

If c is large enough then all objects in cluster C_i are mapped to the interval $[i \cdot c, (i+1) \cdot c)$. The data is stored in a B$^+$-tree according to the *iDistance* values.

Range Query The **Range**(q, r) search algorithm runs separate search procedures for some of the clusters. The cluster C_i is selected for searching if the following condition is satisfied:

$$d(q, p_i) - r \leq \text{max-dist}_i$$

where max-dist$_i$ is the maximal distance between p_i and objects in cluster C_i. Fig. 1 illustrates this condition in two-dimensional space. The cluster searching algorithm examines all objects from C_i which have the *iDistance* value within the interval

$$[d(p_i, q) + i \cdot c - r, d(p_i, q) + i \cdot c + r].$$

The grey areas within the clusters in Fig. 1 represent the space specified by this condition. To examine an object x means to calculate the distance $d(q, x)$ and to add x to the query answer set S if $d(q, x) \leq r$.

***k*-Nearest Neighbor Query** The **kNN**(q, k) algorithm executes a sequence of **Range**(q, r) queries with growing radius r. The condition for adding accessed objects into the answer set differs from the **Range**(q, r) condition. An accessed object x is added into the **kNN**(q, k) answer set S if

$$\text{either} \quad |S| < k \quad \text{or} \quad d(q, x) < d(q, \mathit{farthest}(S, q))$$

where $\mathit{farthest}(S, q)$ is the object from S with the greatest distance from q. The sequence of **Range**(q, r) queries terminates and the **kNN**(q, k) execution is completed when

$$d(\mathit{farthest}(S, q), q) < r \quad \text{and} \quad |S| = k.$$

For each range iteration, the **kNN**(q, k) algorithm stores information about the searched space (taking advantage of the B$^+$-tree properties) in order not to search any space redundantly.

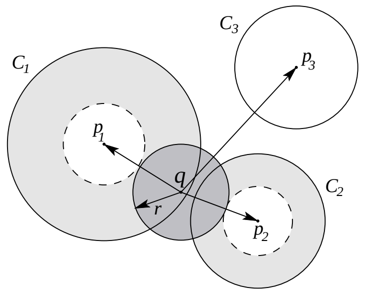


Fig. 1. The *iDistance* search mechanism

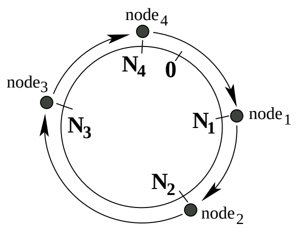


Fig. 2. The *Chord* structure

2.3 Chord

Chord is a purely decentralized structured P2P protocol [10] that provides mechanisms for efficient localization of the node that stores a particular data item (specified by a given search key). It is a message driven dynamic structure that is able to adapt as nodes (cooperating computers) join or leave the system.

The protocol uses *consistent hashing* [11] which uniformly maps the domain of search keys into *Chord* key domain of size 2^m. The parameter m should be large enough to make probability of the hash collisions negligible. Every *Chord* node is assigned a key N_i from the interval $[0, 2^m)$ as well. The node with key N_i is "responsible" for all keys from the interval $(N_{i-1}, N_i]$ (mod 2^m) – see Fig. 2 for visualization. A node stores objects with the keys the node is responsible for.

Every node maintains a routing table called *finger table* which stores physical addresses of up to m other nodes [10]. The node knows physical addresses of its predecessor and successor as well.

Due to the uniformity of the *Chord* key domain distribution, *Chord* structure has the following properties:

- in an n-node system, the node responsible for a given key is located via $\mathcal{O}(\log n)$ number of messages to other nodes (number of hops);
- the load of particular nodes (number of objects stored in the node) is approximately equal for all nodes.

These properties are very important for performance of the data structure proposed in this paper and are referenced below.

3 ChiDistance

In this section, we introduce *chiDistance* – a new distributed data structure and algorithms for similarity search in metric spaces. The system is based on the idea of *iDistance* (see Section 2.2) and extends it as follows:

- *chiDistance* employs metric pivot selecting techniques to generalize the *iDistance* applicability to metric spaces. Having a finite sample data set $S \subseteq \mathbb{U}$, a set of k pivots is selected and then *Voronoi*-like partitioning [2] is used to identify the clusters (see Section 3.1 for details).

- It uses the *Chord* protocol and the *chiDistance* key assignment mechanism to distribute the storage space among arbitrary number of cooperating nodes (Section 3.3).
- It provides distributed search algorithms for the Range and kNN queries that parallelize the time-consuming queries execution (Section 3.4).

As mentioned in Section 2.3, the *Chord* protocol assumes the uniform distribution of the key domain to keep its efficiency. Because *iDistance* distribution is strongly non-uniform, the *iDistance* values must be transformed by a hash function with a uniform distribution. Then the *Chord* protocol can be used to divide the storage space among the cooperating nodes. Section 3.2 describes the mechanism of finding the uniform key transformation.

Both, the pivot selection and the phase of determining the uniform hash function, must proceed during the initialization phase of the algorithm. When pivots $\{p_1, \ldots, p_k\}$ are selected, data set is divided into clusters $\{C_1, \ldots, C_k\}$, and the uniform transformation h is found, the *chiDistance* value of an object $x \in C_i$ is computed as follows:

$$chi(x) = h(d(p_i, x) + i \cdot c). \tag{1}$$

Fig. 3 illustrates the process of the algorithm initialization and *chiDistance* computation.

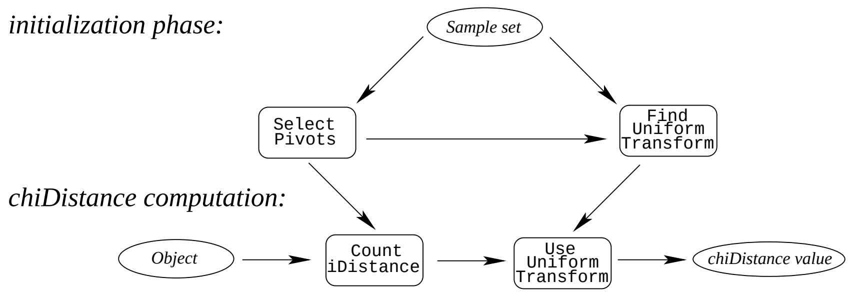


Fig. 3. Algorithm initialization and *chiDistance* computation process

3.1 Pivots Choosing Procedure

As mentioned in Section 2.2, the *iDistance* algorithm uses vector space properties to partition the data space and then to select the reference points (pivots) of identified partitions. In order to generalize the applicability of the method to metric spaces, we use a metric-based technique to efficiently select a set of pivots.

Then, having a set of k pivots $\{p_1, \ldots, p_k\}$, we divide objects from $\mathbb{I}$ (the set of indexed objects) into clusters $C_1, \ldots, C_k$ as follows:

$$C_i = \{x | x \in \mathbb{U}, d(p_i, x) < d(p_j, x), 1 \leq j \leq k, j \neq i\}.$$

This partitioning technique is referred to as Voronoi-like partitioning [2].

Generally, the similarity search algorithms eliminate some data objects from the search process without computing their distances to the query object. For pivot-based data structures, the main objective of finding a suitable set of pivots is to increase the effectiveness of such pruning of the search space.

Let us summarize the **Range**(q, r) search algorithm described in Section 2.2. An object x from cluster C_i (with pivot p_i) is accessed if its *iDistance* value ($iDist(x) = d(p_i, x) + i \cdot c$) is within the interval

$$[d(p_i, q) + i \cdot c - r, d(p_i, q) + i \cdot c + r].$$

This condition can be reformulated as follows: An object $x \in C_i$ can be eliminated without accessing if

$$|d(p_i, x) - d(p_i, q)| > r.$$

Thus, the higher the value $|d(p_i, x) - d(p_i, q)|$ for objects $x \in C_i$, $\forall i \in \{1, \ldots, k\}$ the more effective the search algorithm.

The *chiDistance* pivot selection technique is based on the general technique described in [12] which is tuned to fit our search algorithm. Having a sample set of data objects $S \subseteq \mathbb{U}$, we try to choose a set of k pivots $\{p_1, \ldots, p_k\}$ from S in order to "maximize" the function $|d(p_x, x) - d(p_x, y)|$ for every $x, y \in S$ where $p_x \in \{p_1, \ldots, p_k\}$ is the pivot closest to object x. One way to do this is to maximize the mean of distribution of $|d(p_x, x) - d(p_x, y)|$ on S. Let us denote this mean value as $\mu_{\{p_1, \ldots, p_k\}}$ for set of pivots $\{p_1, \ldots, p_k\}$. Now, we can say that $\{p_1, \ldots, p_k\}$ is a better set of pivots than $\{p'_1, \ldots, p'_k\}$ when

$$\mu_{\{p_1, \ldots, p_k\}} > \mu_{\{p'_1, \ldots, p'_k\}}.$$

The $\mu_{\{p_1, \ldots, p_k\}}$ value is estimated in the following way: a set of n pairs $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ is chosen randomly from the sample set S. For each pair (x_i, y_i) the closest pivot p_{x_i} is found and value $v_i = |d(p_{x_i}, x_i) - d(p_{x_i}, y_i)|$ is computed. The value $\mu_{\{p_1, \ldots, p_k\}}$ is estimated as

$$\mu_{\{p_1, \ldots, p_k\}} = \frac{1}{n} \sum_{i=1}^{n} v_i.$$

We need $k + 1$ distance computations to obtain value v_i for each pair of objects. Therefore, $n \cdot (k + 1)$ distance computations are needed to estimate $\mu_{\{p_1, \ldots, p_k\}}$.

Now, knowing how to compare two sets of pivots, we can define the selection technique itself. We use the incremental algorithm described in [12]. First, pivot p_1 is chosen from a set of s candidates (selected from the sample set S) such that p_1 has the maximum $\mu_{\{p_1\}}$ value. Then, a second pivot p_2 is selected from another set of s objects such that $\mu_{\{p_1,p_2\}}$ is maximal considering p_1 fixed. This process is repeated k-times to get a set of k pivots.

In every iteration of the described algorithm, we can store the closest pivot p_{x_i} to x_i and their distance $d(p_{x_i}, x_i)$ (for every $1 \leq i \leq n$). Thus, only $d(p_j, x)$ and (maybe) $d(p_j, y)$ values need to be computed when the j^{th} pivot is added. Therefore, maximally $2ns$ distance computations are needed while adding a new pivot. Repeating this step k times, the total cost of the pivot selection process is $2kns$ distance computations.

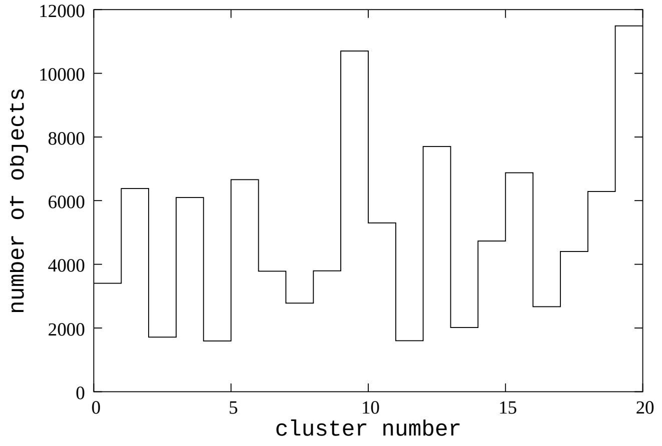


Fig. 4. Number of data objects in clusters

Fig. 4 shows an example of how the data is partitioned to clusters identified by the described method (number of objects in each cluster). The data set is composed of 100,000 three-dimensional vectors (see Section 4 for details about the data set). The sample set consists of 1,000 object pairs ($n = 1000$); number of pivot candidates in every algorithm step is 100 ($s = 100$) and the number of selected pivots is twenty ($k = 20$).

3.2 Uniform Order-Preserving Transformation

The *Chord* routing protocol assumes a uniform layout of the nodes (peers) on the key space circle to keep the property of logarithmic navigation through the structure (Section 2.3). The objective our structure would like to reach is the balanced storage load of the nodes (volume of data stored in the nodes). In order to meet these two criteria, the key domain should have uniform distribution.

The distribution of the *iDistance* domain is strongly non-uniform. Fig. 5 visualizes a typical *iDistance* distribution using 20 pivots. This domain can be

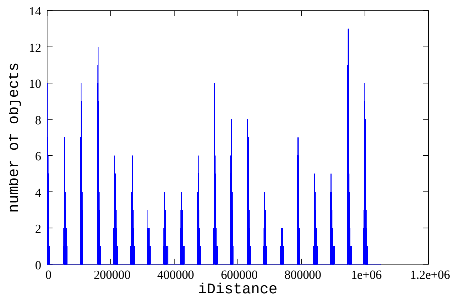


Fig. 5. Distribution of the *iDistance* domain

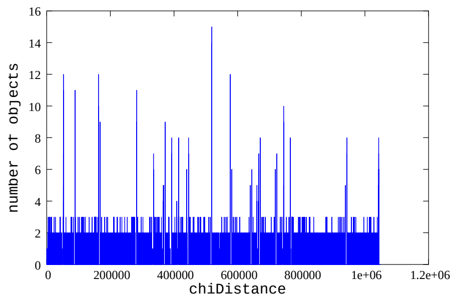


Fig. 6. Distribution of the *chiDistance* domain – h function used

transformed by a hash function with a uniform distribution. Of course, such transformation must be order-preserving to keep *iDistance* properties.

We use a technique described in [13] designed for finding order-preserving uniform transformations between arbitrary domains. Let us denote the desired transformation $h : [0, A] \longrightarrow [0, B]$ (in our case the $[0, A]$ interval is the *iDistance* domain and the $[0, B]$ interval is the *chiDistance* domain of optional size).

First, this method divides target space $[0, B]$ into p intervals of the same length where p is parameter of the algorithm (its precision). This partitioning gives a sequence of values $b_0, b_1, \ldots, b_p$ from $[0, B]$. Then, having an non-decreasing sample sequence $a_1, a_2, \ldots, a_n$ of keys from $[0, A]$, we select the following $p + 1$ values from this sequence:

$$a_0, a_{\lceil \frac{n}{p} \rceil}, \ldots, a_{\lceil i \cdot \frac{n}{p} \rceil}, \ldots, a_n.$$

Let us denote $a_{(i)}$ element $a_{\lceil i \cdot \frac{n}{p} \rceil}$ for every $0 \le i \le p$. The desired transformation h maps $h(a_{(i)}) = b_i$ for every $0 \le i \le p$. These values are fixed and the h values for all other keys from $[0, A]$ are computed as the linear interpolation of them:

$$h(x) = (x - a_{(i-1)}) \cdot \frac{b_i - b_{i-1}}{a_{(i)} - a_{(i-1)}}, \text{ for } x \in (a_{(i-1)}, a_{(i)}).$$

Thus, h is the piecewise-linear transformation. The "quality" of the uniformity of the image domain distribution depends on the precision factor p. Fig. 6 shows the distribution of the domain from Fig. 5 after applying function h. The precision factor $p = 200$; the sample set $\{a_0, a_1, \ldots, a_n\}$ of size $n = 5000$ has been selected randomly from the whole data set of size 100,000. Size of both domains is 2^{20}.

3.3 *chiDistance* Data Structure

The proposed data structure is a message driven structured P2P system based on the *Chord* routing mechanism. Each node of the structure is logically composed of two layers – *Chord* layer and *chiDistance* layer. The topology of the network is given by the *Chord* protocol (the *Chord* layers of the nodes communicate with each other to form and maintain the structure).

When a new node wants to *join* the structure, it contacts an already participating node. It is assigned a key from the *chiDistance* domain and it receives data objects with keys from the interval of its responsibility (Section 2.3). The keys for nodes are chosen uniformly at random. The joining node receives the "*chiDistance* configuration" as well – the selected pivots and the uniform transformation h. To *leave* the system, a node notifies its successor node and sends all stored data to it.

The *chiDistance* layer forms the interface of the system on every node. When receiving an *insert/delete/exact-match* operation request, first, it calculates the *chiDistance* value for the object passed by the operation (Equation 1). Then the *Chord* layer locates the node that is responsible for the computed key and the operation request is passed to that node to store/delete/get the object.

Due to usage of the constant c in the *chiDistance* key formula (1), the domain consists of separated segments that correspond to particular clusters. It may happen that one node is responsible for intervals of keys belonging to several clusters. Vice versa, the interval corresponding to a cluster can be divided among several adjacent *chiDistance* nodes.

Every node stores the data separately for every covered cluster. The data objects are stored in a Red-Black tree based structure that provides guaranteed $\log(n)$ time cost for *get*, *put* and *remove* operations and provides *range*(*from*, *to*) operation as well. This storage policy is convinient for the search algorithm.

3.4 Range Search Algorithms

The *chiDistance* **Range**(q, r) search algorithm follows the basics of *iDistance* algorithm and distributes it parallelizing the query execution. The query processing starts on the initiating node and then spreads over the other nodes.

The algorithm searches the clusters $C_1, C_2, \ldots, C_k$ separately. The segment of data to be searched within each cluster C_i is dependent on the *chiDistance* key that would be assigned to object q if object q was in cluster C_i:

$$chi_i(q) = h(d(p_i, q) + i \cdot c).$$

This value is computed for every cluster C_i and the requests for clusters search are sent to the nodes that are responsible for these chi_i keys – let us denote these nodes N_i.

Within cluster C_i, all objects with *chiDistance* keys from the following interval must be examined:

$$[h(d(p_i, q) + i \cdot c - r), h(d(p_i, q) + i \cdot c + r)].$$

Thus, the N_i node explores all data objects from this interval that are stored in N_i. But objects from cluster C_i can be stored in several adjacent nodes so – if N_i is not responsible for the terminal points of this interval – the search request is forwarded to the predecessor and/or successor node(s) of N_i. In order to parallelize the execution, node N_i first forwards the requests and then searches its own data space.

For every accessed data object x, the distance $d(q, x)$ is computed and if $d(q, x) \leq r$ then x is added to the $\mathbf{Range}(q, r)$ query answer set S. The partial answer sets from all visited nodes are returned to the query originating node and are joined together to form the final anwer set of the range query.

4 Performance Evaluation

In this section, we present results of experiments we conducted on our prototype implementation of *chiDistance* data structure. The system forms a logical overlay network independent of the physical location of the participating nodes. The communication among the nodes is realized via messages using the UDP and TCP communication protocols.

4.1 The Data Sets and Parameter Settings

We executed our experiments on two data sets. The first is a set of artificially generated three-dimensional vectors of real numbers with the L_2 (Euclidian) metric distance function (VEC). This data set has a uniform distribution of distances between objects and all objects have the same size.

The second data set is a real-life set of sentences from the Czech national corpus with the *edit distance* function as the metric (TXT). The length of the sentences varies significantly and the distribution of the distances is rather skewed – most of the distances are within a small range of values.

Both data sets consist of 100,000 objects. All experiments are performed on a structure formed by up to 100 cooperating nodes running on PCs connected by a high-speed local network. The first executed process is given a sample data set of 5,000 objects randomly selected from the whole data set. This sample set is used during the pivot selection (Section 3.1) and then to determine the uniform hash function h (Section 3.2). The number of pivots/clusters is fixed to $k = 20$ – exploring the influence of this parameter to the performance is part of our future work. The precision parameter p of the hash function h is set to 200 which seems to be sufficient for the size of the used *chiDistance* domain $[0, 2^{20})$.

4.2 Domain Coverage and Load Factor

Every node joining the system is assigned a key from the *chiDistance* domain and covers a domain segment $(\mathit{predecessor}, \mathit{node}]$ (Section 3.3). Fig. 7 shows the size of intervals covered by each of 100 nodes (the domain size is $2^{20} = 1048576$).

One of the characteristics influencing the system performance is how the data set is distributed among the nodes. Fig. 8 shows number of data objects stored in each of the nodes (load factor). As discussed in Section 3.2, the distribution of the *chiDistance* domain is broadly uniform and therefore the load factor rather copies the domain coverage from Fig. 7.

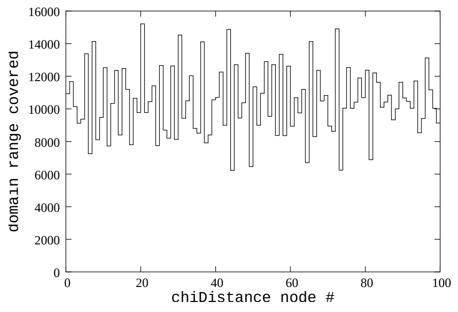


Fig. 7. Segments of *chiDistance* domain covered by individual nodes

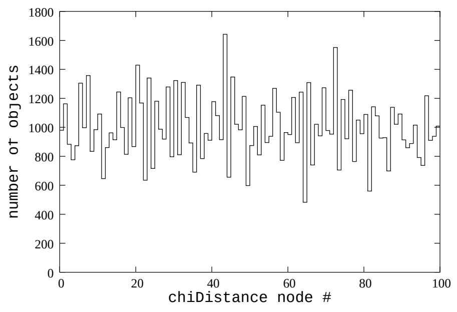


Fig. 8. Number of data objects stored by individual nodes

4.3 Key Locating – Insert, Delete, Exact-match

The *Chord* routing mechanism used by our system is able to locate the node responsible for given *chiDistance* key in $\mathcal{O}(\log n)$ number of hops (where n is the number of nodes in the system). The key locating mechanism is used by most of the operations on the structure and its complexity is very important. Fig. 9 shows the experimental results – the hop count with respect to the number of nodes in the system (n). The values were obtained as an average over 100 operations.

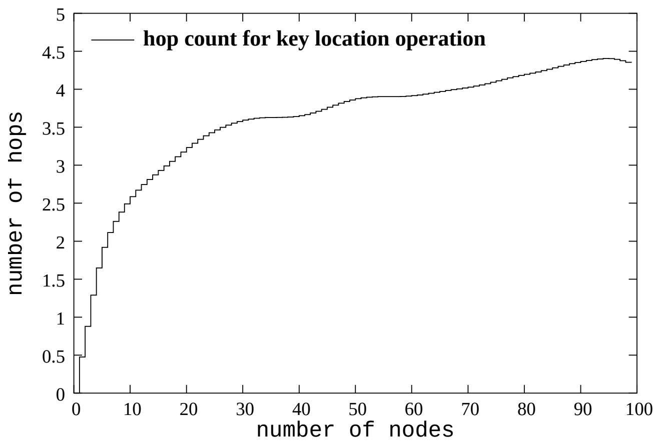


Fig. 9. Number of hops to locate the node responsible for given *chiDistance* key.

The complexity of the key localization is the complexity of the *insert*, *delete*, and *exact-match* operations as well.

4.4 Range Search Performance

In this section, we present results of experiments analyzing the performance of the **Range**(q, r) search algorithm. To measure the computational cost of the operation, we use the number of distance computations on the participating nodes. This performance metric is common for similarity search algorithms because the actual cost of the distance computation may differ signifiantly for various distance functions. In the experiments, we have neglected the inter-nodes communication time because the distance computations are more time consuming than sending a message between nodes.

Thus, most of the analyses in this section are based on observing how the distance computations are distributed on the set of nodes (under various circumstances). The first observation is that the total number of distance computations (*total cost*) remains the same for any number of nodes as well as for the centralized *iDistance* method. This quantity predicates generally about this search method efficiency with respect to our pivot choosing technique.

One of the very important contributions of distributed structures is the parallelism of a query execution. In our experiments, we observe several aspects of the *intraquery* and *interquery* parallelism [14]. We quantify the intraquery parallelism as the parallel computational cost of one query execution – maximal number of distance computations performed in sequence (the *parallel cost*).

Fig. 10 presents results of the experiment which measured the parallel cost with respect to the growing search radius. The second curve in the graph shows the total number of distance computations (divided by 10 to see both shapes properly). All values presented in this section are taken as an average over 100 queries with different query objects randomly chosen from the data set.

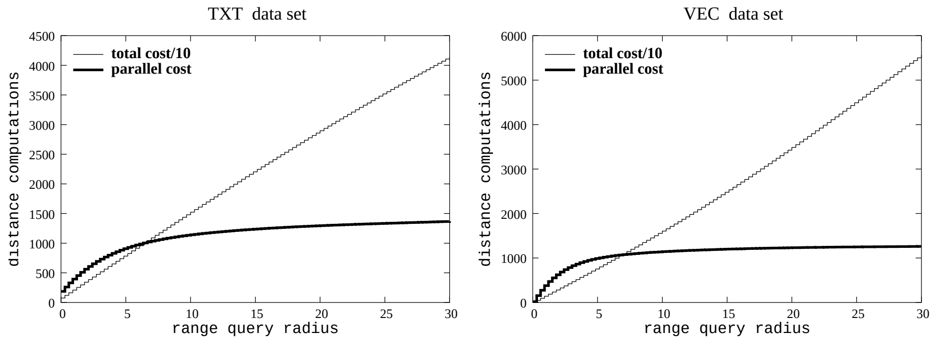


Fig. 10. The total and parallel cost for range queries increasing the radius.

This experiment results show that, for larger query radii, the parallel cost corresponds to the nodes load factor – some nodes access all the data objects they store. Let us denote C_q the cluster which the query object belongs to. The search algorithm principles imply that (for larger radii) most of the objects in cluster C_q must be accessed. Very likely, some node(s) store(s) only objects from cluster C_q because, in our setting, the number of nodes is higher than the number of pivots/clusters. This causes the rapid increase of the parallel cost. Obviously, most of the nodes are computationally loaded significantly less than the parallel cost is, because the average cost per server (total cost/100) is lower. This fact is analyzed by the interquery parallelism experiments.

The interquery parallelism refers to the ability of the system to accept multiple queries at the same time. Let us denote the set of active nodes $\{n_1, \ldots, n_m\}$ and the numbers of distance computations on node n_i as $cost_i^1, cost_i^2, \ldots, cost_i^k$ for the sequence of k queries. We measure the interquery parallelism through summation of the number of distance computations over a sequence of range queries (on each node separately):

$$\textit{inter-cost}_i = \sum_{j=0}^{k} cost_i^j.$$

Maximizing this value over the set of nodes $\{n_1, \ldots, n_m\}$ we get a total parallel cost of the sequence of k queries:

$$\textit{inter-cost} = \max\{\textit{inter-cost}_1, \ldots, \textit{inter-cost}_m\}.$$

Fig. 11 shows the *inter-cost* value for growing number of parallel queries (k) and for selected radii. The query objects were chosen at random and the graph values are taken as an average over 10 executions of different sets of k queries. As discussed above, mainly the nodes storing objects from cluster C_q are significantly computationally loaded during the **Range**(q, r) execution. Thus, the *inter-cost* is significantly smaller than k-multiple of the parallel cost of one range query (with corresponding radius).

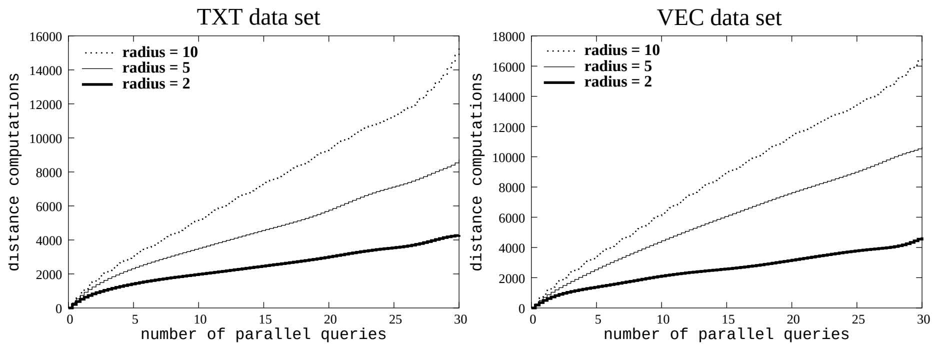


Fig. 11. The interquery parallelism for growing number of parallel queries

5 Conclusions and Future Work

So far, the field of distributed index structures for metric data has not been much investigated. We consider this topic very relevant for needs of the preset-day applications processing large volumes of digital data. We propose *chiDistance*, a distributed data structure for similarity search in metric spaces. The structure is based on the idea of *iDistance* index method. The applicability of *iDistance* is generalized from vector spaces to metric spaces by a pivot selection technique tuned in order to fit the *chiDistance* search algorithm. A Peer-to-Peer structure based on the *Chord* protocol is created to distribute the storage space and to parallelize the execution of similarity queries.

We present results of performance experiments conducted on our prototype implementation and on two data sets. Among other things, we have studied several aspects of parallelism of the range search algorithm. The results show that the computational cost on participating nodes varies but the cost on a single node is always upper bounded by the number of data objects stored by the node. Recall that the data is quite uniformly distributed among the nodes. At the same time, very good interquery parallelism is achieved when processing a set of range queries in parallel.

Our future work will concern the $\mathbf{kNN}(q, k)$ search algorithm. As well, we want to study the influence of number of pivots to the search algorithms performance. In this paper, we have considered the presented structure rather static by using fixed quantity of sources (nodes) regardless of the volume of stored data. In future, we want to study the scalability of the system by adding nodes as the volume of stored data increases. Finally, we plan to conduct tests that would compare the performance of GHT* [7] and *chiDistance* – both implemented over the same network infrastructure. We see this as a unique opportunity to compare structures and algorithms under the very same conditions.

References

1. Hjaltason, G.R., Samet, H.: Index-driven similarity search in metric spaces. ACM Trans. Database Syst. **28** (2003) 517–580
2. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. ACM Comput. Surv. **33** (2001) 273–321
3. Koudas, N., Faloutsos, C., Kamel, I.: Declustering spatial databases on a multi-computer architecture. In: EDBT. (1996) 592–614
4. Gupta, A., Agrawal, D., El Abbadi, A.: Approximate range selection queries in peer-to-peer systems. In: Proceedings of the First Biennial Conference on Innovative Data Systems Research, Asilomar, California, United States (2003)
5. Tanin, E., Harwood, A., Samet, H.: A distributed quadtree index for peer-to-peer settings. In: ICDE. (2005)
6. Banaei-Kashani, F., Shahabi, C.: Swam: a family of access methods for similarity-search in peer-to-peer data networks. In: CIKM '04: Proceedings of the Thirteenth ACM conference on Information and knowledge management, ACM Press (2004) 304–313

7. Batko, M., Gennaro, C., Zezula, P.: Scalable similarity search in metric spaces. In: Proceedings of the DELOS Workshop on Digital Library Architectures: Peer-to-Peer, Grid, and Service-Orientation, Edizioni Libreria Progetto, Padova (2004) 213–224
8. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer file sharing technologies (2002)
9. Yu, C., Ooi, B.C., Tan, K.L., Jagadish, H.V.: Indexing the distance: An efficient method to KNN processing. In: VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy, Morgan Kaufmann (2001) 421–430
10. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of ACM SIGCOMM, ACM Press (2001) 149–160
11. Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., Lewin, D.: Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web. In: STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, ACM Press (1997) 654–663
12. Bustos, B., Navarro, G., Chávez, E.: Pivot selection techniques for proximity searching in metric spaces. In: SCCC 2001, Proceedings of the XXI Conference of the ChileanComputer Science Society, IEEE CS Press (2001) 33–40
13. Garg, A.K., Gotlieb, C.C.: Order-preserving key transformations. ACM Trans. Database Syst. **11** (1986) 213–234
14. Özsu, M.T., Valduriez, P.: Distributed and parallel database systems. ACM Comput. Surv. **28** (1996) 125–128

Query Tuning through Refinement/Enlargement in a Formal Context

Carlo Meghini
CNR – ISTI, Pisa, Italy
meghini@isti.cnr.it

Nicolas Spyratos
Universitè de Paris Sud – LRI
spyratos@lri.fr

1 Introduction

The work reported in this paper originates in the following basic observation: the user of an information system rarely knows exactly what he is looking for, but once shown a piece of information he can quickly tell whether it is what he needs.

Query tuning is the process of searching for the query that best approximates the information need of the user. We note that, finding the query that best expresses a given information need is important not only for retrieving the information satisfying the current need, but also in order to name and store the query for use at some later time (without having to re-invent it over and over).

In advanced information systems, query tuning proceeds in two steps: (a) the user navigates the information space until he finds a subspace of interest and (b) in that subspace, the user issues a query. If the answer to the query is satisfactory then the session terminates, otherwise a new navigation step begins. However, navigation and querying are two completely separate processes, and the user usually has to switch often from one to the other -a painstaking process usually producing a frustrating experience.

In this paper, we propose an approach to query tuning that interleaves, or integrates navigation and querying into a single process, thus leading to a more flexible and more user friendly method of query tuning.

The proposed approach is based on formal concept analysis [2, 1], and models the directory of a digital library as a formal context in which the objects represent documents and the attributes represent indexing terms [3]. As a result, the concepts of the underlying concept lattice represent meaningful classes of documents, in the sense that all documents in a class share the same set of indexing terms. Therefore, we propose to use the concept lattice as the basic navigation tool.

We assume the user queries to be Boolean combinations of indexing terms, and more specifically conjunctions of indexing terms. Indeed, the objective of this paper is not to propose a new, more powerful query language but, rather, use an existing (simple) language in order to illustrate our approach to query tuning.

Our approach is user-controlled, and proceeds as a sequence of refine/enlarge commands until a user-approved query is obtained. More precisely, query tuning in our approach proceeds roughly as follows:

- *Query mode* The user formulates a query to the library and receives an answer; if the answer is satisfactory then the session terminates and the issued query is considered tuned, otherwise the user can issue a Refine or an Enlarge command.

- *Refine* A refine command returns all maximal concepts (from the concept lattice) that refine the user query (in the sense that their extent is included in the answer received by the user);

then the user can decide to either return to query mode and query one or more of those concepts, or terminate.

- *Enlarge* An enlarge command returns all minimal concepts (from the concept lattice) that subsume the user query (in the sense that their extent includes the answer received by the user); then the user can decide to either return to query mode and query one or more of those concepts, or terminate.

In what follows, we first introduce briefly the digital library model that we use, in section 2, and then we recall the basic notions needed from formal concept analysis, in section 3. Our query tuning approach is presented in section 4, illustrated through a running example. Finally, we offer some concluding remarks and outline a research agenda, in section 5.

2 Digital Libraries

A *digital library* serves a network of providers willing to share their documents with other providers and/or consumers (hereafter, collectively called "users"). Each document resides at the local repository of its provider, so all providers' repositories, collectively, can be seen as a distributed repository of documents spread over the network. The digital library acts as a mediator, supporting transparent access to all sharable documents by the library users. Two of the basic services supported by the library are *registration* and *querying.*

Registration When a provider wishes to make a document sharable over the network of users he must register it at the library. To do so he must provide two items to the library:

- the document identifier
- the document description

We assume that the document identifier is a global identifier, such as a URI, or just the URL where the document can be accessed, (however, for convenience of notation, we use integers in our examples). As for the document description, we consider only content description and we assume that such a description is given by selecting a set of terms from a controlled vocabulary. For example, the document description $\{\mathsf{QuickSort}, \mathsf{Java}\}$ would indicate that the document in question is about the quick sort algorithm written in Java.

Therefore, to register a document, its provider submits to the library an identifier i and a set of terms D. We assume that registration of the document by the library is done by storing a pair (i, t) in the library repository, for each term t in D. In our previous example, if i is the document identifier, the library will store two pairs: $(\mathsf{i}, \mathsf{QuickSort})$ and $(\mathsf{i}, \mathsf{Java})$. The set of all such pairs (i, t) is what we call the *library directory,* or simply *directory* (the well known Open Directory (`http://dmoz.org/about.html`) is an example of such a directory). Clearly the directory is a binary relation between document identifiers and terms, *i.e.*a formal context (in the sense defined in the next section).

Querying Library users access the library in search of documents of interest, either to use them directly (*e.g.*, as learning objects) or to reuse them as components in new documents that they intend to compose. Search for documents of interest is done by issuing queries to the library, and the library uses its directory to return the identifiers (*i.e.*, the URIs) of all documents satisfying the query.

The query language that we use is a simple language in which a query is just a Boolean combination of terms:

$$q ::= t \mid q_1 \wedge q_2 \mid q_1 \vee q_2 \mid q_1 \wedge \neg q_2 \mid (q)$$

where t is any term.

The answer to a query q is defined recursively as follows:

```
if $q$ is a term then $ans(q) = \{i \mid (i, q)$ is in the directory$\}$
else begin
    if $q$ is $q_1 \wedge q_2$ then $ans(q) = ans(q_1) \cap ans(q_2)$
    if $q$ is $q_1 \vee q_2$ then $ans(q) = ans(q_1) \cup ans(q_2)$
    if $q$ is $q_1 \wedge \neg q_2$ then $ans(q) = ans(q_1) \setminus ans(q_2)$
    end
```

In other words, to answer a query, the underlying digital library management system simply replaces each term appearing in the query by its extension from the directory, and performs the set theoretic operations corresponding to the Boolean connectives.

3 Formal Concept Analysis

Let $\mathcal{O}$ be a set of objects and $\mathcal{A}$ a set of attributes. A formal context, or simply context over $\mathcal{O}$ and $\mathcal{A}$ is a triple $(\mathcal{O}, \mathcal{A}, C)$ where $C \subseteq \mathcal{O} \times \mathcal{A}$ is a binary relation between $\mathcal{O}$ and $\mathcal{A}$. Figure 1 shows a formal context in tabular form, in which objects correspond to rows and attributes correspond to columns. The pair (o, a) is in C (that is, object o has attribute a) if and only if there is a x in the row corresponding to object o, in the column corresponding to attribute a.

	A	B	C	D	E	F
1	x		x	x	x	
2		x	x			
3	x		x	x		x
4		x	x	x	x	x
5	x	x			x	x

Figure 1: A Formal Context

Let i and e be respectively the functions *intension* and *extension* as they are normally used in information systems, that is:

$$\begin{aligned} i(o) &= \{a \in \mathcal{A} \mid (o, a) \in C\} \text{ for all } o \in \mathcal{O} \\ e(a) &= \{o \in \mathcal{O} \mid (o, a) \in C\} \text{ for all } a \in \mathcal{A}. \end{aligned}$$

In Figure 1, the intension of an object are the attributes marked with a x in the row corresponding to the object; the extension of an attribute are the objects marked with a x in the attribute column. Now define:

$$\begin{aligned} \varphi(O) &= \bigcap\{i(o) \mid o \in O\} \text{ for all } O \subseteq \mathcal{O} \\ \psi(A) &= \bigcap\{e(a) \mid a \in A\} \text{ for all } A \subseteq \mathcal{A} \end{aligned}$$

A pair (O, A), $O \subseteq \mathcal{O}$ and $A \subseteq \mathcal{A}$, is a *formal concept* of $(\mathcal{O}, \mathcal{A}, C)$ if and only if $O = \psi(A)$ and $A = \varphi(O)$. O is called the *extent* and A the *intent* of concept (O, A). In the formal context shown in Figure 1, $(\{1, 3, 4\}, \{C, D\})$ is a concept, while $(\{1, 3\}, \{A, D\})$ is not.

The concepts of a given context are naturally ordered by the *subconcept-superconcept* relation defined by:

$$(O_1, A_1) \leq (O_2, A_2) \text{ iff } O_1 \subseteq O_2 (\text{iff } A_2 \subseteq A_1)$$

This relation induces a lattice on the set of the concepts of a context. For instance, the concept lattice corresponding to the context shown in Figure 1 is presented in Figure 2, where for convenience some concepts are explicitly indicated (in red).

There is an easy way to "read" the extent and intent of every concept from the lattice. To this end, we define two functions γ and μ, mapping respectively objects and attributes into concepts, as follows:

$$\begin{array}{rcll} \gamma(o) & = & (\psi(\varphi(\{o\})), \varphi(\{o\})) & \text{for all } o \in \mathcal{O} \\ \mu(a) & = & (\psi(\{a\}), \varphi(\psi(\{a\}))) & \text{for all } a \in \mathcal{A}. \end{array}$$

It is easy to see that $\gamma(o)$ and $\mu(a)$ are indeed concepts. In addition, $(o, a) \in C$ is equivalent to $\gamma(o) \leq \mu(a)$. The functions γ and μ are represented in Figure 2 by labeling the concept $\gamma(o)$ with o as a subscript, and $\mu(a)$ with a as a superscript. Thus, concept 14 is indeed $\mu(A)$, while concept 13 is $\gamma(2)$. Finally, it can be proved that for any concept $c = (O_c, A_c)$, we have:

$$\begin{array}{rcl} O_c & = & \{o \in \mathcal{O} \mid \gamma(o) \leq c\} \\ A_c & = & \{a \in \mathcal{A} \mid c \leq \mu(a)\}. \end{array}$$

Thus, the extent of concept 12 is given by $\{1, 3, 4\}$ since $\gamma(1)$, $\gamma(3)$ and $\gamma(4)$ are all and the only concepts smaller than concept 12; the intent of this concept is $\{C, D\}$ since it is smaller than both $\mu(C)$ (concept 12 itself) and $\mu(D)$.

It is interesting to note that the conjunction of all terms in the intent of a concept is actually a minimal conjunctive query whose answer is the extent of the concept ("minimal" here means that if we omit any of the terms in the intent we no longer obtain the extent as the answer). Conversely, any answer to a conjunctive query is the extent of some concept.

It follows that, by navigating the concept lattice, the user can be guided to the best result he can obtain with a conjunctive query. These facts and observations lie at the heart of our query tuning approach that we present in the next section.

4 Query Tuning

During user interaction with the digital library, query tuning is obtained by using four commands: *Query, Terminate, Refine* and *Enlarge.* In this section we define each of these commands separately and illustrate them using a running example. We recall that we restrict our attention to conjunctive queries only.

Query

The user issues a query to the system. The system returns the answer, obtained by evaluating $\psi(A)$ where A is the set of terms in the query. In fact:

$$ans(\bigwedge A) \quad = \quad \bigcap \{ans(a) \mid a \in A\} \quad \text{(by definition of } ans \text{ on conjunctions)}$$

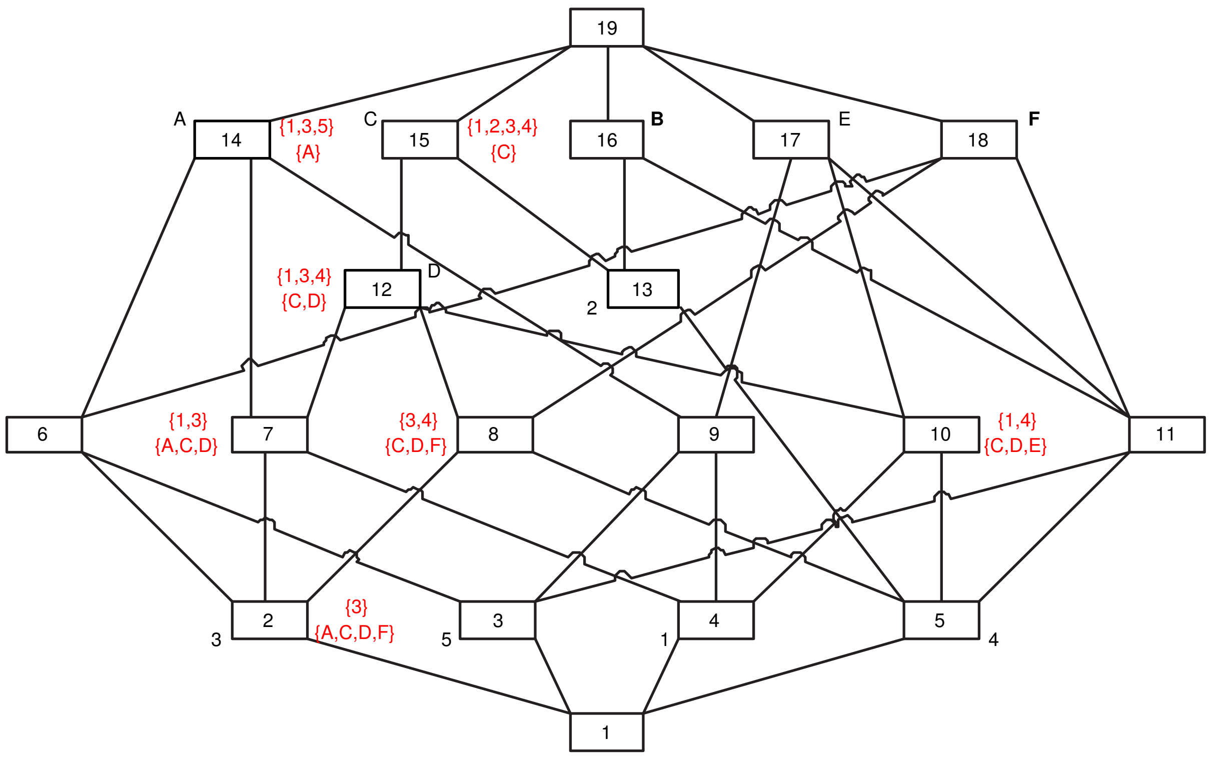


Figure 2: A concept lattice

$$
\begin{aligned}
&= \bigcap\{\{o \in \mathcal{O} \mid (o,a) \in C\} \mid a \in A\} \quad \text{(by definition of } \mathit{ans} \text{ on terms)}\\
&= \bigcap\{e(a) \mid a \in A\} \quad \text{(by definition of } e\text{)}\\
&= \psi(A) \quad \text{(by definition of } \psi\text{).}
\end{aligned}
$$

The system also shows to the user the most precise, *i.e.*most specific query that can be used to obtain the same result, given by the conjunction of the terms in $\varphi(\psi(A))$. In other words, upon evaluating a query $\bigwedge A$ the system "places" itself on the concept $(\psi(A), \varphi(\psi(A)))$, which becomes the current concept. Notice that this concept can be computed from the query in polynomial time in the size of the context.

In our example, let us assume the user poses the query D. In response, the system returns the answer $\psi(D) = \{1,3,4\}$ and shows the query $C \wedge D$, since $\varphi(\psi(D)) = \{C, D\}$. The current concept is therefore concept 12 in Figure 2.

Terminate

The user is satisfied by the answer and issues a *Terminate* command. The system switches to next-query mode. Otherwise, the user performs one of the actions described next.

Refine

The user judges the answer to be too rich, *e.g.*the cardinality of the answer set is too big or, upon inspection, there are too many irrelevant answers in the answer set; and issues a *Refine* command. The system then returns two pieces of information to the user:

1. The intent A_{max} of each maximal concept max whose extent O_{max} is strictly contained in the current answer; and
2. The objects in the extent of the current concept lying outside O_{max}.

Let us explain further these two items that are returned to the user:

1. Any concept like max above is a maximal sub-concept of the current concept, whose intent is by definition a superset of the current concept intent. The system computes these concepts and shows their intents to the user, by simply presenting him the additional terms of each intent with respect to the current concept. This computation can be done in polynomial time with respect to the size of the context . Let us see how in our example. Let us assume that the current answer $\{1, 3, 4\}$ is too large for the user, who executes a *Refine.* To compute the terms to be shown to the user, we look at a smaller context, consisting of the objects in the extent of the current concept, and of the attributes outside the intent of the current concept. The context we are looking at is:

	A	B	E	F
1	×		×	
3	×			×
4		×	×	×

   In order to achieve maximality, from this context we select the terms with maximal extension, that is A, E and F. Each of these terms t must be shown to the user, since it leads to a maximal sub-concept of the current concept, given by:

   $$(\psi(A_c \cup \{t\}), \varphi(\psi(A_c \cup \{t\})))$$

   where A_c is the intent of the current concept. In our example, term A leads to concept 7 $(\{1, 3\}, \{A, C, D\})$, term E leads to concept 10 $(\{1, 4\}, \{C, D, E\})$, term F leads to concept 8 $(\{3, 4\}, \{C, D, F\})$, which are all maximal sub-concepts of the current concept, *i.e.*12. Notice that the query associated to each such concepts is a refinement of the query originally posed by the user.

2. The second item shown to the user is the set of objects which are "lost" by selecting the corresponding refinement. For instance, along with the term F, the user is shown the object set $\{1\}$ containing the answers which are no longer such if the query is refined by adding the term F to it.

The complete answer that the user gets in response to a *Refine* in our example is reported in Table 1. We recall that the current concept is $(\{1, 3, 4\}, \{C, D\})$. For convenience, the table also shows the query refinement and the concept corresponding to each solution. Upon deciding whether to accept a proposed refinement, the user can figure out the attributes he gains by inspecting the added terms, or the answers he looses by inspecting the lost objects. If the user does decide to move, then the concept corresponding to the selected solution becomes the current concept, and a new interaction cycle begins (by querying, refining and enlarging).

Notice that if no maximal sub-concept exists, *i.e.*the current concept is the least concept of the lattice, then the system returns empty and, subsequently, the user may issue an *Enlarge* command (see below) or try a new query.

Solution	Added Terms	Lost Objects	Query Refinement	Concept
1	A	4	$A \wedge C \wedge D$	$(\{1,3\},\{A,C,D\})$
2	E	3	$C \wedge D \wedge E$	$(\{1,4\},\{C,D,E\})$
3	F	1	$C \wedge D \wedge F$	$(\{3,4\},\{C,D,F\})$

Table 1: Result of a *Refine*

Enlarge

The user judges the answer to be too poor (*e.g.*, the cardinality of the answer set is too small, possibly zero), and issues an *Enlarge* command. The system then returns two pieces of information to the user (in a similar manner as in the case of *Refine*):

1. The intent A_{min} of each minimal concept min whose extent O_{min} strictly contains the current answer. Each such concepts min is a minimal super-concept of the current concept, and the set of such concepts can be computed in polynomial time in an analogous way to the maximal sub-concepts. Let us again see how in our example. Let us assume that the user refines the initial query by selecting solution 1 in Table 1, and that he then asks to enlarge this set. To compute the objects leading to a minimal super-concept of the current concept, we look at a smaller context, consisting of just the attributes in the intent of the current context and of the objects outside the extent of the current context. That is:

	A	C	D
2		x	
4		x	x
5	x		

From this context we select the objects with maximal intention, that is 4 and 5. Each of these objects o leads to a minimal super-concept of the current concept, given by:

$$(\psi(\varphi(O_c \cup \{o\})), \varphi(O_c \cup \{o\}))$$

where O_c is the extent of the current concept. In our example, object 4 leads back to concept 12 $(\{1,3,4\},\{C,D\})$ while object 5 leads to concept 14 $(\{1,3,5\},\{A\})$, which are all minimal super-concepts of the current concept, *i.e.*7. Notice that the query associated to each such concepts is a relaxation of the query associated to the current concept.

2. The objects in O_{min} which lay outside the extent of the current context.

The complete answer that the user gets in response to an *Enlarge* in our example is reported in Table 2. We recall that the current concept is $(\{1,3\},\{A,C,D\})$. For each alternative solution, the Table shows the terms that are lost in the query relaxation, the added objects, the relaxed query and the corresponding concept. Upon deciding whether to accept a proposed enlargement, the user can figure out the attributes he looses or the answers he gains. If the user does decide to move, then the concept corresponding to the selected solution becomes the current concept, and a new interaction cycle begins (by querying, refining and enlarging).

Notice that if no minimal super-concept exists, *i.e.*the current concept is the greatest concept of the lattice, then the system returns empty and, subsequently, the user may issue a *Refine* command or try a new query.

Solution	Lost Terms	Added Objects	Query Refinement	Concept
1	A	4	$C \wedge D$	$(\{1,3,4\},\{C,D\})$
2	C,D	5	A	$(\{1,3,5\},\{A\})$

Table 2: Result of an *Enlarge*

5 Concluding remarks

We have seen an approach to query tuning that combines navigation and querying into a single process thus providing a more flexible and more user friendly interaction between the users and the information system.

In the traditional approach, the interaction proceeds by repeating the following two steps (in some order): (1) Query and Terminate, or (2) Navigate. In our approach, the interaction proceeds by repeating the following three steps (in some order) before terminating: (1) Query, (2) Refine, or (3) Enlarge. Here, Refine and Enlarge represent navigation steps that might be interleaved with the Query step and might be repeated several times before termination, *e.g.*, Query-Enlarge-Query-Refine-Query-Enlarge- ... -Terminate.

In order to keep the presentation as simple as possible, we have considered only conjunctive queries. However, as our approach works with extents, what we have said holds not only for conjunctive queries but also for queries with negation and/or disjunction.

References

[1] B.A. Davey and H.A. Priestley. *Introduction to lattices and order*, chapter 3. Cambridge, second edition, 2002.

[2] B. Ganter and R. Wille. Applied lattice theory: Formal concept analysis. `http://www.math.tu.dresden.de/~ganter/psfiles/concept.ps`.

[3] P. Rigaux and N. Spyratos. Metadata inference for document retrieval in a distributed repository. In *Proceedings of ASIAN'04, 9th Asian Computing Science Conference*, number 3321 in LNCS, Chiang-Mai, Thailand, 8-10 December 2004. Invited Paper.

I know I stored it somewhere - Contextual Information and Ranking on our Desktop

Wolfgang Nejdl and Raluca Paiu

L3S Research Center and University of Hannover
Deutscher Pavillon, Expo Plaza 1, 30539 Hannover, Germany
{nejdl,paiu}@l3s.de

1 Motivation

Future digital libraries will be distributed, and recent research has already explored some promising approaches focusing on distributed and peer-to-peer search and retrieval architectures, connecting distributed repositories efficiently and transparently. Another aspect, which has been less explored so far, is the role of the implicit personal repositories we all have on our desktops, and the efficiency we expect to gain from storing important resources in these "personal digital libraries". Ironically, in many cases, moving important resources closer to our workspace results in less retrieval efficiency.

Sophisticated web search technology usually allows us to find appropriate documents in a few seconds. Finding these documents on our desktop is surprisingly more difficult, at least if we have been storing documents for a few years or more. This is improving somewhat with the recent crop of desktop search engines, but even with these tools, searching through our (relatively small set of) personal documents with the recent beta of Google Desktop Search is inferior to searching the (rather vast set of) documents on the web with Google. The main reason for this is that one of the distinguishing features of Google - sophisticated ranking using PageRank and other features - is unavailable on our desktop. This position paper explores some of the information we have available on our desktop to extend search beyond simple full-text search, and the algorithms we can build upon this information, implementing efficient ranking of resources on our desktop.

Regarding the first aspect, we propose activity-based metadata and relationships as sources of additional information in desktop search. This information in many cases represents contextual information, which is very useful for re-finding resources we already worked with. We will describe corresponding metadata for two selected scenarios. For the second aspect, we will focus on recent advances of PageRank-based ranking, extending models such as the ones describing our contextual information, and show how local and global ranking measures can be integrated in such a model.

These two techniques together show considerable promise for extending efficient information access to our desktop, extending globally available information with user-centered activity-based information, and exploiting the unique information background we have available on our desktop. We are currently implementing first prototypes in the context of the open source Beagle project which aims to provide sophisticated desktop search in Linux.

2 Representing Context Information

Available desktop contexts. Current desktop search prototypes fall short of utilizing desktop specific information, especially context information. Two of these missed opportunities include:

Email context. Documents sent as attachments lose all contextual information as soon as they are stored on the PC, even though emails usually include additional information about their attachments, such as sender, subject or valuable comments. We might discuss a paper with a colleague during a brainstorming session, and then afterwards send her the electronic version via email, together with a few helpful comments. After a while, our colleague might not remember details about the paper itself, but rather recall with whom she discussed it or which question was raised in the discussion and included as comment in the email or email thread. We would like to find the stored paper not only based on its content, but also associatively based on that context information.

Browsing context. Browser caches include all information about user's browsing behaviour, which are useful both for finding relevant results, and for providing additional context for results. When searching for a document we downloaded from the CiteSeer repository, we would like to retrieve not only the specific document, but also all the referenced and referring papers which we downloaded on that occasion as well.

Documents stored from emails and from web sites form our personal digital library, which holds the papers we are interested in. We should store contextual information for these documents to retrieve them efficiently, and to restore the original context we built up when storing these documents. Personalized ranking on the desktop should take this contextual information into account as well as the preferences implicit in this information. [4] discusses how people tend to associate things to certain contexts. So far, however, search infrastructures neither collect nor use this contextual information.

Scenario specific annotation ontologies. Figure 1 presents our current prototype ontology, which specifies context metadata for emails, files, web pages and publications, together with the relations among them, described in more detail in [2]. Conceptually, the elements in the rectangles represent classes, circles represent class attributes. We use classes whenever we want to attach importance/rank on entities, attributes otherwise.

Metadata for the email context refer to the date an email was sent or accessed, the subject of the email and the email body. Emails are connected to their attachments through the 1:n *has_attachment* relation. The *reply_to* relation represents thread information. *Person* represents the sender of an email. Another important relationship represents the connection between emails / email attachments and the documents they are saved as, as we want to use all attributes originally connected to the email a document was attached to when we want to find it again.

For the browsing context, we annotate each page with additional information about its basic properties (URL, access date, etc), as well as more complex ones such as in- and out-going links browsed. An extended publication ontology makes use of additional knowledge about how CiteSeer pages are connected and what they represent. Publications are referenced by other publications and can cite others, they can have a publication date / year associated with them, as well as a conference or journal. Publications have authors and are stored as documents on the desktop.

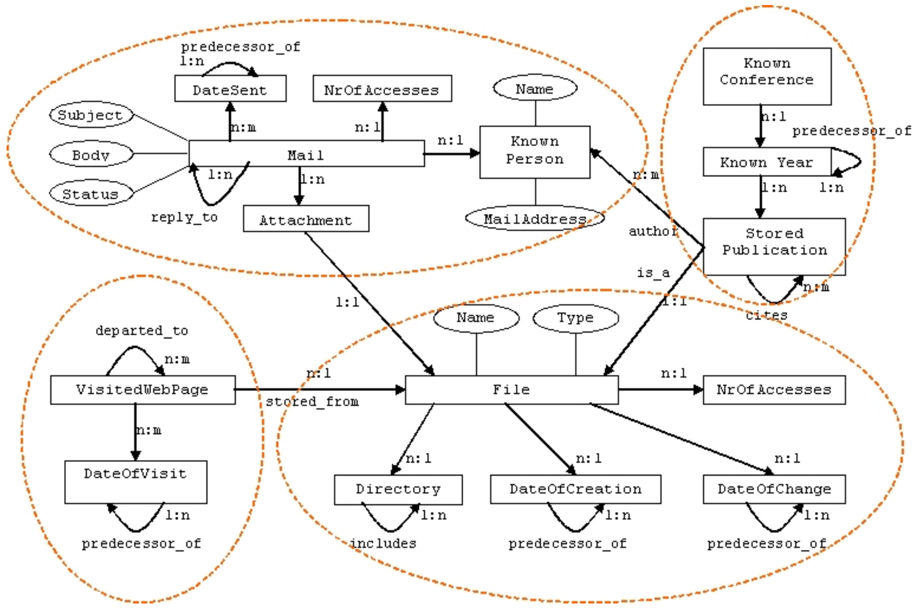


Fig. 1. Context ontology for our prototype

Other ontologies describe contexts like conferences, including reviewers, papers, meetings, authors, or private contexts like birthdays, including persons, locations, etc.

3 Authority Transfer and Ranking

Authority transfer annotations. Annotation ontologies should describe all aspects and relationships among resources, which influence the ranking. The identity of the authors for example influences our opinion of documents so “author” should be represented explicitly as a class in our publication ontology. Second, we have to specify how these aspects influence each others importance.

ObjectRank [1] has introduced the notion of authority transfer schema graphs, which extend schemas similar to the ontologies previously described, by adding weights and edges in order to express how importance propagates among the entities and resources inside the ontology. These weights and edges represent the authority transfer annotations, which extend our context ontologies with the information we need to compute ranks for all instances of the classes defined in the context ontologies.[1]

Figure 2 depicts our context ontology plus its authority transfer annotations. For example, authority of an email is split among the sender of the email, its attachment, the number of times that email was accessed, the date when it was sent and the email to which it was replied. So, if an email is important, the sender might be an important person, the attachment an important one and/or the number of times the email was

[1] In contrast to ObjectRank, we do not compute a keyword-specific ranking, but a global one.

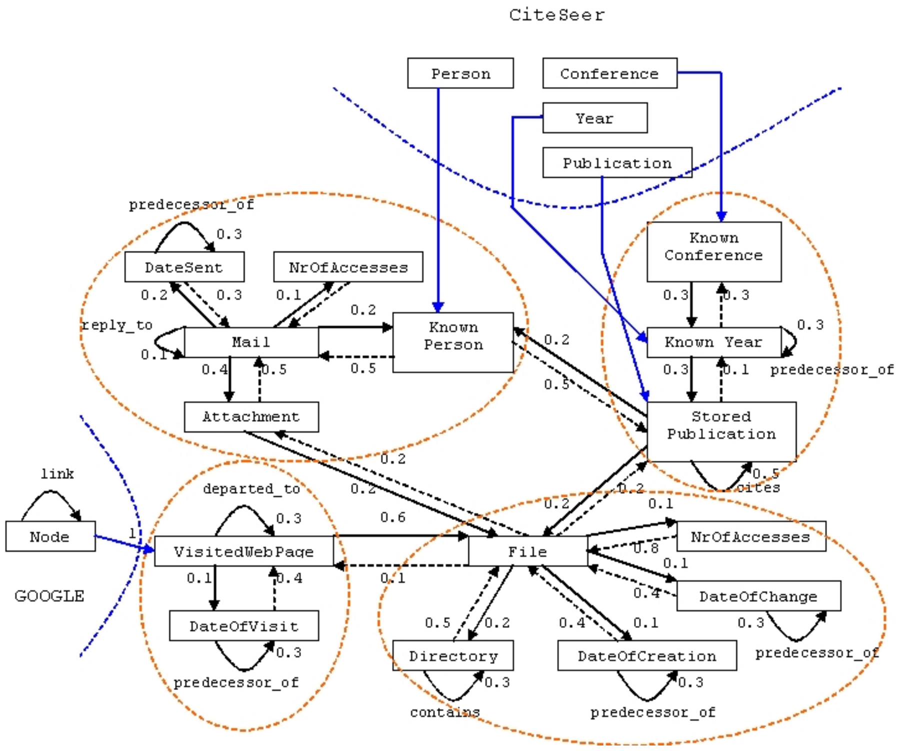


Fig. 2. Authority transfer annotations, including external ranking sources

accessed is very high. Additionally, the date when the email was sent and the previous email in the thread hierarchy also become important. As suggested in [1], every edge from the schema graph is split into two edges, one for each direction. This is motivated by the observation that authority potentially flows in both directions and not only in the direction that appears in the schema (if we know that a particular person is important, we also want to have all emails we receive from this person ranked higher). The final ObjectRank value for each resource is calculated based on the PageRank formula.

The ontology representing our web browsing context says that a visited webpage is important if we arrived at the current one from an important page, if the file under which it is stored is important, or if the date when the page was visited is important. For the CiteSeer context, publications transfer part of their authority to other papers they cite, to their authors, to the files under which they are stored, and to the year when the paper was published. As we can see, citing important papers doesn't make a paper important.

Personalized Preferences and Ranking. Different authority transfer weights express different preferences of the user, translating into personalized ranking. The important requirement for doing this successfully is that we include in a users ontology all concepts, which influence her ranking function. For example, if we view a publication important because it was written by an author important to us, we have to represent that in our context ontology. Another example are digital photos, whose importance is usually heavily influenced by the event or the location where they were taken. In this case both event and location have to be included as classes in our context ontology.

Personal vs. External Ranking. For the computation of authority transfer, we can also include additional external ranking sources, connecting global ranking computation and personalized ranking of resources on our desktop. These external ranking sources are used to provide the seed values for the calculation of the personal ranking. Our prototype ontology includes two global ranking services, one returning Google ranks, the second one ranks computed from the CiteSeer database.

4 Conclusions and Related Work

This paper has explored two techniques - activity-based metadata and authority transfer annotations - as important contributions towards enabling efficient retrieval and ranking for the "personal digital repositories" building up on our computers. Activity-based metadata describe context information relevant for finding and connecting the resources we store on our desktop, authority transfer annotations help to rank retrieved resources in a personalized way. Global ranking services like Google or Citeseer-derived ranking services can initialize these personalized ranking measures. Our prototype uses the open source project Beagle[2] as underlying desktop search infrastructure and extends its regular full-text indexing capabilities with contextual metadata and ranking.

Facilitating search for information the user has already seen before is also the main goal of the *Stuff I've Seen (SIS)* system, presented in [3]. Based on the fact that the user has already seen the information, contextual cues such as time, author, thubnails and previews can be used to search for and present information. [3] mainly focuses on experiments investigating the general usefulness of this approach, though, without presenting more technical details. Based on SIS, [5] proposes a timeline-based visualization of search results over personal content. This basic timeline view is then augmented with public (holidays, news headlines) and personal (calendar appointments and digital photographs) landmark events. The main goal of this system is to facilitate browsing.

References

1. A. Balmin, V. Hristidis, and Y. Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *VLDB*, Toronto, September 2004.
2. P. Chirita, R. Gavriloaie, S. Ghita, W. Nejdl, and R. Paiu. Activity based metadata for semantic desktop search. Technical report, L3S, December 2004.
3. S. Dumais, E. Cutrell, JJ Cadiz, G. Jancke, R. Sarin, and Daniel C. Robbins. Stuff i've seen: A system for personal information retrieval and re-use. In *SIGIR*, Toronto, July 2003.
4. Teevan J., Alvarado C., Ackerman M. S., and Karger D. R. The perfect search engine is not enough: A study of orienteering behavior in directed search. In *CHI*, Vienna, April 2004.
5. M. Ringel, E. Cutrell, S. Dumais, and E. Horvitz. Milestones in time: The value of landmarks in retrieving information from personal stores. In *INTERACT*, Zurich, September 2003.

[2] http://www.gnome.org/projects/beagle/