Searching very large bodies of data using a transparent peer-to-peer proxy

Mike Taylor
Index Data UK
48A Carysfort Road
London N8 8RB
mike@indexdata.com
www.indexdata.com

Marc Cromme
Index Data DK
Købmagergade 43
1150 Copenhagen K
marc@indexdata.dk
www.indexdata.dk

Abstract

While individual data stores are increasingly large, the aggregate size of the Internet dwarfs them all and always will. We consider an approach to searching rich documents across a very large network of individual data stores using a transparent peer-to-peer proxy. This approach is dependent on the use of a standardised search-and-retrieve protocol sufficiently rich to enable semantics to be induced on both its documents and its queries. Candidate protocols include the mature Z39.50 and the more recent SRW/U, of which the latter is considered more "web-friendly". Networks of the peers underlying this approach to large-repository search and retrieval may take on widely differing topologies, and queries may be routed in widely different ways. Optimal values of tuning parameters may be determined using an evolutionary system in which simulations of different configurations compete against each other. The European collaborative project Alvis is using the approach outlined in this paper to build a semantic peer-to-peer search engine aggregated across multiple subject-specific repositories. Among the problems still to be solved, the matter of how to merge results from multiple peers is the most difficult.

1. Introduction

The ever-increasing availability and ever-decreasing cost of mass storage hardware enables organisations and even individuals to maintain increasingly huge compendia of data on the computer systems that they are responsible for. However, we should not be too proud of these individual technological wonders we've created: the ability to store terrabytes of information in any one repository is insignificant compared with the power of the Internet. By combining multiple repositories across the Internet, it is possible to form data stores that exceed the those of the largest system by many orders of magnitude. This paper outlines an approach that leverages the use of established semantically rich Internet protocols to gain access to large distributed bodies of data via a transparent peer-to-peer network. We describe the approach in a series of incremental steps.

2. Standardised semantically rich search-and-retrieve protocol

We begin by considering the advantages of using a standardised search-and-retrieve protocol that is rich enough to have a semantic interpretation induced on it. Although HTTP is standardised and can be used for searching and retrieval, on its own it does not meet our need because there are no standard semantics attached to the URLs requested, the data POSTed or the documents returned. What is needed is a protocol that, like HTTP, is well enough defined to be independently implemented by clients and servers with confidence that they will interoperate, but which also specifies standard semantics for structured queries and responses.

At present, there are two realistic contender protocols for this role: ANSI/NISO Z39.50 [12] and SRW/U [6]. While either Z39.50 or SRW/U would be suitable for the architecture we outline in this paper, we concentrate on SRW/U, due to the likely faster uptake of this standard.

2.1 Z39.50

Z39.50 is a mature and powerful standard initially developed under the auspices of the American standards bodies ANSI and NISO. In 1998, it was ratified by ISO as international standard ISO 23950, but it is still known mostly by the older name. The standard is maintained by an agency sponsored by the Library of Congress. The data structures transmitted in Z39.50 dialogues are expressed using ASN.1 and encoded using BER - technologies which are considered either antiquated or mature in various communities.

Historically, Z39.50 has been most extensively deployed in the library world, as a means of locating and viewing MARC calalogue records. However, it has also been used

in domains as diverse as government information, geospatial metadata, chemical formulae and navigating hierarchical thesauri.

Simple object-oriented APIs for building Z39.50 clients are available as free implementations in most of the major programming languages [14].

2.2 SRW/U

The use of ASN.1 and BER in Z39.50 is widely perceived - whether rightly or not - as mounting a barrier to implementation. In response to this perception, work began in December 2000 on recasting the powerful and expressive Z39.50 semantics in terms of mechanism more readily understood in the contemporary information environment. The result of this effort is a family of two new protocols: SRW (the Search/Retrieve Web-service) and SRU (Search/Retrieve URL). Their specifications are administered by the SRW Editorial Board, which is made up eleven members from institutions including commercial companies, libraries and universities.

SRW uses SOAP to deliver structured query payloads from client to server, and responses including zero or more records from server to client. The requests and responses are both expressed in XML. Several operations are defined, each consisting of a request-response pair. These including "searchRetrieve", "explain" (in which a client asks a server to describe its capabilities) and "scan" (for browsing index entries). The specification for a related Update operation is under development [13] and will be released during 2005.

SRU is semantically equivalent to SRW, but uses a simpler, REST-like, mechanism as its transport. SRU requests are expressed as URLs with query parameters that carry information equivalent to that in the corresponding SRW-request XML documents. SRU response payloads are identical to those of SRW, but are returned directly as the content of the HTTP response rather than being wrapped in a SOAP envelope as in SRW.

Note that SRW and SRU are both based on HTTP, and can be legitimately seen as profiles of that protocol. They represent a refinement of, and greater precision in, what information is transmitted over HTTP and how that information is interpreted. Thus, while HTTP is not itself suitable for IR, it is a very suitable substrate for a protocol that facilitates rich IR.

Although SRW was initially seen as the more suitable vehicle for serious IR usage, uptake of SRU has been quicker and broader. SRU implementations seem in general to outperform related SRW implementations due to the additional XML-parsing overhead in the SOAP-based protocol.

A very important aspect of structured search-and-retrieve protocols such as Z39.50 and the SRW/U family is the provision of a structured query language in which rich queries can by expressed. Z39.50 uses an esoteric binary query format known as the "Type-1 query", which need not concern us here. SRW/U improves on its predecessor by using a textual query format which is at once intuitively understandable and powerfully expressive - capable of represeting all the queries that can be expressed in Z39.50's Type-1 notation. This format is known as CQL [5] (Common Query Language), and is exemplified by queries such as the following:

```
dinosaur
title=dinosaur
title=(dinosaur or pterosaur)
     and author=martil
dc.title=*saur and dc.author=martill
title exact "the complete dinosaur"
     and date < 2000
name=/phonetic "smith"
fish prox/distance<3/unit=sentence frog
```

This query language allows substantially greater precision than the more traditional type of IR query consisting only of a "bag of words" not related in any specific way or tied at any particular part of the documents being searched.

3. Transparent protocol proxy

The Z39.50 and SRW/U protocols can be proxied, just like HTTP. At present there are not many proxy programs available for these protocols, although this seems likely to change. One such is the YAZ Proxy [15]. Because the Z39.50 and SRW/U protocols that it handles are so much richer than HTTP, it can perform a wider range of services related to to those protocols. For example, an installation of YAZ Proxy at the Library of Congress [8] enhances the library's Z39.50 service in several ways: it increases performance for clients and reduces server load by caching and reusing initialised server sessions; it throttles access to server resources, introducing delays to reduce the frequency with which a single client's requests reach the back end; it protects the server from malformed queries that cause crashes, by recognising them and sending an appropriate diagnostic back to the client; and it generates detailed logs.

4. Fan-out proxy

One of the more ambitious services that a transparent proxy could provide is to "fan out" searches to more than one server, gathering and merging the various servers' results and returning them to the client. In this way, meta-searching may be achieved without the need for the client to know anything about either the process or the configuration. This approach is advantageous because the provision of such a

fan-out proxy allows existing client applications to be easily reconfigured to metasearch without requiring any code changes.

Because proxies appear as servers to clients and as clients to servers, it follows that proxies may by chained together, each one in the chain seeing its neighbours as a client and a server. For fan-out proxies, this chaining capability is very powerful, giving rise to trees rather than chains of servers. A naive Z39.50 or SRW/U client can search a server A; that server may in fact be a fan-out proxy that searches server M, N and O; server N may in turn also be a fan-out proxy, searching servers X, Y and Z. As a result, the client searches all the servers M, O, X, Y and Z without needing to know anything about the topology below it.

There is no effective limit to the depth with which fan-out proxies can search further proxies, so that an arbitrarily "bushy" tree topology can be achieved.

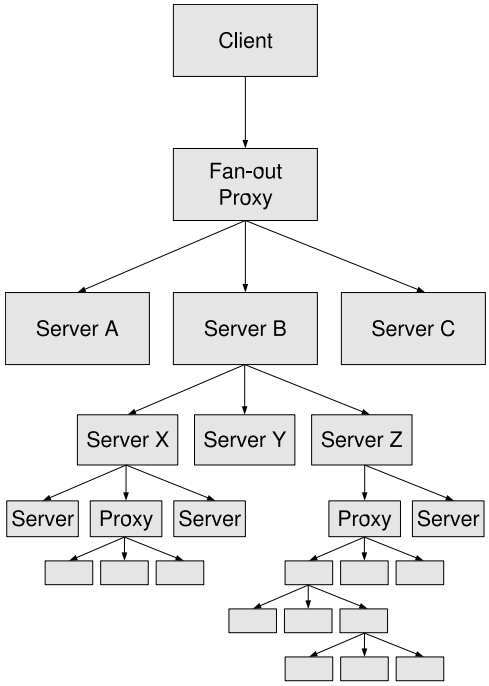

Figure 1: Complex topology with multiple fan-out proxies arranged in a tree.

In principle, then, fan-out proxies are useful and powerful. In practice, however, they are not used because they raise too many administrative problems. Each proxy's administrators need to maintain a database of known servers - a problem that grows proportionally more troublesome as the total size of the network increases, since loop detection becomes more difficult. In consequence, a tree of fan-out proxies does not scale well, and so is an unsatisfactory solution to the problem of searching in very large distributed repositories. A different approach is needed.

5. Peer-to-peer proxy

Our preferred approach, then, introduces a different kind of proxy: a loose federation of nodes forming a peer-to-peer network. Each peer in the network may function on behalf of a client (as a way of injecting a search into the network), or of a server (as way of executing the query against a suitable individual repository). Peers may also represent both a client and a server ("servent") as in Gnutella [9] and similar protocols; and it also possible for a peer to be a part of a network while not associated with any client or server, purely to contribute to the routing of queries.

Peers pass queries between themselves using their own dedicated protocol, separate from the client-server protocol; it is the task of the network as a whole to find the most appropriate servers to answer each query, and to this end, peers continually discover more information about each other. The topology of the peer network, then, is dynamic - connections between peers are constantly being replumbed to connect peers that can benefit more fully from communication with each other.

From the perspective of both client and server, the peer network appears as a single, simple proxy: all the complexity of the network is internal to it, concealed in a "cloud" from the applications that use it.

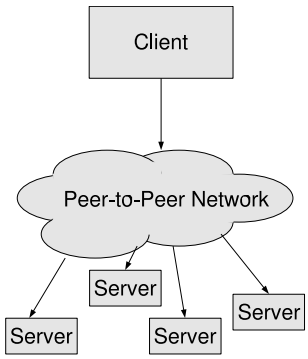

Figure 2: Peer-to-peer topology with peers communicating between themselves behind the scenes. The entire peer-to-peer network masquerades as a server to its clients, and a a client to all the servers.

6. Operation of the peer-to-peer network

A peer-to-peer network proxy allows the administrative problems of the fan-out tree to be solved by automated means. Cruicially, no peer requires a human administrator to maintain a catalogue of servers to search. Instead, each peer dynamically maintains a pool of "neighbour" peers that it knows about. Neighbours which seem not to be very useful (e.g. because they do not forward many queries,

or do not provide useful responses to the queries they are asked to answer) may be dropped from the pool; and new neighbours may be discovered via interactions with existing neighbours, and be added to the pool.

An important consequence of this approach is that a new peer can join the network without any administrator needing to take action: instead, the peer connects to any peer that is already in the network, and through that peer quickly finds out about useful neighbours; and it, in turn, is discovered by peers that were on the network before it.

The problem of searching within the network can be seen from the perspective of the query itself, wandering between peers in search of a server that can answer it. The main job of a peer is to assist the query on its quest, routing it to its most relevant neighbours. The assessment of a neighbour's relevance to a query can be done using relatively primitive mechanism such as checking each neighbour's previous record in dealing with queries that contain the same or similar leaf terms. More sophisticated mechanisms can be introduced when additional information is available, for example if a peer advertises what subject areas it covers.

Suppose a particular peer, Peer A, has to handle a query and does so by passing it to a neighbour, Peer B. Peer B may not be able to handle the query directly, but has another neighbour, Peer G, that can. In this case, the information passed back from B to A includes details about Peer G, so that A can add it to its own pool of neighbours if appropriate.

It is wasteful if a query is ever forwarded back to a peer that it has already passed through. Detecting and avoiding this kind of loop is difficult using a tree of fan-out proxies, but easy in the peer-to-peer network proxy: this aspect of query routing is handled at the protocol level, using facilities that do not exist in client-server search-and-retrieve protocols such as SRW/U.

As a query travels through the network, it carries with it a count of how many more peers it is allowed to go through, to ensure that its journey terminates. This is called the query's Time To Live, or TTL. When a given peer is handling the query, it can allocate the remaining TTL in various ways: the query might be forwarded to a single neighbour with all its TTL; it might be forwarded to two equally promising neighbours, with each forwarded query receiving half of the TTL; or it might be passed to several neighbours with the remaining TTL allocated in proportion to how relevant the destinations seems to the query. We envisage that different peers will use different forwarding strategies, and that this "ecological diversity" will help to make the network robust.

Queries may carry with them an indication of what forwarding strategy they prefer. Peers associated with clients may use the facility to create "tracer bullet" queries, which make many branchless hops before finally fanning out broadly using their last few hops. Such "long, thin" query trajectories provide a means for peers to periodically probe remote parts of the network in order to discover new relevant neighbours.

Frameworks have been proposed for obtaining semantic interoperability among data sources in a bottom-up, semi-automatic manner without relying on pre-existing, global semantic models [1, 11]. Semantic alignment of query interpretation between peers in the network is greatly facilitated by the use of CQL, because this language makes use of "context sets", which rigorously define the semantics of the indexes they provide for use in queries. For example, the index-name "dc.title" indicates the "title" index as defined in the "dc" context set, which means title in the Dublin Core sense, as opposed to the land registry sense or the heraldry sense. This precision allows searching within well-defined and unambiguous semantic categories which are guaranteed to be understood the same way by both client and server.

7. Using peer-to-peer proxies in ALVIS

The ongoing European collaborative project ALVIS [4, 7] is using the approach outlined in this paper to build a semantic peer-to-peer search engine aggregated across multiple subject-specific repositories. The complete ALVIS system includes semantic components beyond the scope of this paper, including linguistic analysis in multiple languages, the use of subject-specific taxonomies, and various applications of relevance. In ALVIS, the peer-to-peer approach described in this paper is elaborated by a second, lighter weight peer-to-peer network, which uses distributed hash tables to pre-analyse queries and so to find a suitable peer to use as the query's entry point into the system.

The P2P-IR architecture [2, 3] developed in ALVIS is 5-layered, where the lowest three layers implement a common key based Distributed Hash Table (DHT).

On top of that layer 4 implements the query-adaptive [10] update procedures needed to keep the DHT in good working order to route structured semantic queries to the peers capabile of serving suitable documents.

Finally, layer 5 is the query processing at the local peer, in our case represented by a Z39.50 or SRW/U enabled data storage node, which is able to resolve CQL queries to ranked hit lists and to retrieve the indicated documents. It is implemented using the Zebra Z39.50/SRW/U server and semi-structured text and XML indexer [16].

In ALVIS terminology, the layer 5 capable peers are termed "superpeers" or "fat peers", and those making up the overlay network (layer 1-4) are known simply as "peers".

Within individual ALVIS superpeers, subject-specific searching is enriched by the use of linguistic techniques to break apart documents, mark up parts of speech and assess relevance to various subject areas, the better to perform the final matching of queries to documents. Because these re-

finements are brough to bear within individual peers, they do not affect the whole-system topology.

8. Conclusion

This new approach to searching large distributed repositories is very promising for at least three reasons: because its decentralisation aids scalability; because it removes the need for costly manual administration; and because it separates the large-repository spanning work from that of the client and server software, which can therefore be existing off-the-shelf products such as the Zebra.

However, much work remains to be done. Perhaps the largest area of difficulty is that of merging results from multiple sources into a single result set, as each peer must do that forwards a query to more than one neighbour. Such merging should eliminate duplicate results, and canonicalise relevance scores.

The protocol to be used within the peer-to-peer proxy has been designed but not yet deployed in a large network. As a part of the ALVIS research, we intend to run simulations of large networks (tens of thousands of nodes), with each simulation using a different set of parameters to control query-routing behaviour - for example, initial TTL, likelihood of forwarding a query to multiple neighbours rather than picking the single most promising one, frequency of "tracer bullet" queries, readiness to discard infrequently used peers from the pool of neighbours. By comparing the quality of search results obtained during runs of each candidate configuration, these paramaters can be progressively refined to yield the best possible performance.

We hope ultimately to see an extremely large network of peers running in the real world, with thousands of institutions and individuals adding peers to the network to increase the total size of the distributed repository available to searching clients.

References

[1] Karl Aberer, Philippe Cudre-Mauroux, and Manfred Hauswirth. Start making sense: The chatty web approach for global semantic agreements. Technical report, Ecole Polytechnique Federale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland, 2003.

[2] Karl Aberer, Fabius Klemm, Martin Rajman, and Jie Wu. An architecture for P2P information retrieval. In *27th Annual International ACM SIGIR Conference (SIGIR 2004), Workshop on Peer-to-Peer Information Retrieval*, Sheffield, UK, July 2004.

[3] Karl Aberer and Jie Wu. Towards a common framework for peer-to-peer web retrieval. In Matthias Hemmje, editor, *From Integrated Publication and Information Systems to Virtual Information and Knowledge Environments*. Springer LNCS, EJN-Festschrift, November 2004.

[4] ALVIS - superpeer semantic search engine. EC FP6 Programme Project IST-1-002068-STP. http://www.alvis.info/.

[5] SRW Editorial Board. CQL - common query language, 2004. http://www.loc.gov/z3950/agency/zing/cql/.

[6] SRW Editorial Board. SRW - search/retrieve web service, 2004. http://www.loc.gov/z3950/agency/zing/srw/.

[7] Wray L. Buntine and Michael P. Taylor. ALVIS: Superpeer semantic search engine. In *European Workshop on the Integration of Knowledge, Semantic and Digital Media*, Royal Statistical Society, London, November 2004.

[8] Larry E. Dixson. YAZ++ enhancements coming soon. http://www.indexdata.dk/pipermail/yazlist/2004-March/000857.html.

[9] Patrick Kirk. Gnutella file sharing and distribution network, 2003. http://rfc-gnutella.sourceforge.net/.

[10] Fabius Klemm, Anwitaman Datta, and Karl Aberer. A query-adaptive partial distributed hash table for peer-to-peer systems. In *International Workshop on Peer-to-Peer Computing and DataBases (P2P&DB 2004)*, Crete, Greece, March 2004.

[11] Wolfgang Nejdl, Martin Wolpers, Wolf Siberski, Christoph Schmitz, Mario Schlosser, Ingo Brunkhorst, and Alexander Loser. Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In *WWW2003*, Budapest, Hungary, May 2003.

[12] Library of Congress. The Z39.50 maintenance agency. Protocol overview web-page, 2005. http://lcweb.loc.gov/z3950/agency/.

[13] Rob Sanderson. Record update. http://srw.cheshire3.org/docs/update/.

[14] Mike Taylor. ZOOM: The Z39.50 object-orientation model, 2001. http://zoom.z3950.org/.

[15] YAZ Proxy. Open source implementation of a highly configurable Z30.50 and SRW/U proxy. http://www.indexdata.dk/yazproxy/.

[16] Zebra. Open source text and XML indexer software and Z39.50 server. http://www.indexdata.dk/zebra/.