An Overlay-Network Approach for Distributed Access to SRS

Thomas Fuhrmann
Institut für Telematik
Universität Karlsruhe (TH)
Germany

Andrea Schafferhans
Lion Bioscience AG
Heidelberg, Germany

Thure Etzold
Lion Bioscience Ltd.
Cambridge, UK

Abstract

SRS is a widely used system for integrating biological databases. Currently, SRS relies only on locally provided copies of these databases. In this paper we propose a mechanism that also allows the seamless integration of remote databases. To this end, our proposed mechanism splits the existing SRS functionality into two components and adds a third component that enables us to employ peer-to-peer computing techniques to create optimized overlay-networks within which database queries can efficiently be routed. As an additional benefit, this mechanism also reduces the administration effort that would be needed with a conventional approach using replicated databases.

Index terms — Biological Information Retrieval, SRS, Overlay-network Formation, Peer-to-peer Computing

1 Introduction

The increasing amount and complexity of biological data (e.g. generated by genomics projects) makes it more and more difficult to access and analyse the data. SRS is the most widely used data integration system for biological, biochemical and biomedical databases. It enables users of all backgrounds to intuitively access data and permits internal data to be merged with data from the public domain. [4, 3, 2] The most prominent public server at EBI (http://srs.ebi.ac.uk) currently holds more than 130 biological databases. [9, 8]

A key problem with the current structure of SRS is that it is only designed for accessing local databases. This requires the SRS administrators to

- provide local copies of all the databases requried by the site's SRS users, and to
- keep these local copies continuously up to date.

In this paper we propose a mechanism that solves both issues with an efficient and scalable approach. The key idea is to build two overlay networks per database: One tree-like overlay replicates the root copy of the database to all its mirror sites. The second overlay relies on peer-to-peer topology-formation mechanisms to create an efficient search infrastructure on all SRS installations, those that do and those that do not mirror the specific database.

Typically, SRS gives access to many different databases and - even more important - links all these databases. We will show that our proposed mechanism retains this capability at no extra effort, i.e. it yields a distributed and fully scalable version of SRS.

2 Related work

Overlay-network formation has been widely studied recently. Most applications focus on distributed storage and retrieval, a topic that has been spawned by the epidemic spread of Napster and Gnutella. Many more elaborate mechanisms have been proposed since: CAN, Pastry, and Tapestry [6, 7, 10] are some of the examples in this field, sometimes sloppily called distributed hash-table approach. CAN places storage nodes in a virtual d-dimensional cartesian space, the coordinates of which are the hash of the key that is to be searched. Questions can then be routed within this virtual space towards the respective node. Pastry and Tapestry similarly place the nodes on a virtual ring which can be searched with $\log n$ hops.

Integration of distributed data sources for genomic sequence data and annotations is currently being implemented in the *biodas* project [1]. The DAS (distributed annotation system) consists of a reference sequence server, and one or more annotation servers. Annotation servers are specialized for returning lists of annotations across a certain region of the genome. Annotations have an ID that is unique to the server and a structured description that describes its nature and attributes. In order to retrieve annotations from the DAS, reference and annotations servers have to be known to the client.

DATABANKS, a meta-database of publicly available databases, was introduced in SRS version 5.1. [5] This

Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03)
0--7695-1919-9/03 $17.00 © 2003 IEEE

database is generated automatically by retrieving information provided by the individual SRS administrators from all known SRS servers. By querying this meta-database a user can find accessible servers among which she can chose one to connect to.

Unlike the mechanism proposed in this paper, these approaches do not deal with the overlay-network aspects contained in a potential distributed approach, let alone with actively shaping the resulting overlay to yield optimized query latencies, or load-balancing and resilience properties.

3 Problem statement

The original design of SRS aimed at searching and linking local (copies of) databases of biological data, e.g., sequence data. Since typically these databases were flat text files compiled by different research groups, SRS's main focus was handling these files in an efficient manner. [3, 2] To this end, SRS employs an index generation method that also reflects the fact that different databases refer to entries in other databases to supplement their information content (called "linking" in SRS).

Building these indices and subsequently querying the databases requires the local presence of all of the databases at the respective SRS site. Even more, this index building process has to be repeated after each change in any of the database files that were used for an index.

This approach is inappropriate for a world of distributed computing, where local changes and additions to a knowledge database should become quickly available for all systems using this database, and where processing power and storage capabilities are spread over a large number of relatively cheap workstation-sized computers.

Accordingly, we have pursued the following design goal: A distributed SRS system should be able to include databases from any site in the world, keep the system up-to-date at the same pace these databases are updated without requiring any manual interaction at the sites that rely on such databases. Secondly, we wanted to enable simple access to rarely used databases that are not worth to be stored locally.

At the same time, our design is of course required to match the business pattern of the pharmaceutical industry and information providers using SRS today, i.e. it has to provide means for private databases and it must not reveal search patterns to outsiders for not giving, e.g., competitors clues about current research directions.

The following sections describe our proposed mechanism to accomplish these goals.

4 General Architecture Overview

We assume the following basic architectural elements of the distributed SRS the details of which we need not (and cannot) reveal at this point. The key principle is a split of the existing SRS functionality into two components (database access engine and query front-end) and the addition of a third component (query relay nodes).

- The database access engine is a lightweight SRS component that handles queries and accordingly retrieves data from a local database. It employs SRS's techniques to efficiently extract the requested data. In cases where so-called links need to be followed, it also generates the necessary further requests to other databases.
- The query frontend contains the user interface. In addition to a classical SRS installation, it also provides the means to attach bundles of external databases. For the access to restricted databases additional cryptographic authentication credentials need to be provided.
- Query relay nodes contain a genuine extension to the classical SRS in that these software components neither generate inital requests messages representing a query entered by a user, nor do these nodes respond to request messages as a database access engine does. Instead, they only relay request messagess either to other relay nodes or to a database access engine that can respond to the request. In network terminology, these nodes route so-called anycast request messages through the SRS overlay-network.

Request messages contain a unique database identifier, a description of the query, and, if provided, the necessary credentials to access the database.

The following section describes the overlay-network through which these request messages are routed.

5 Overlay-formation Process

The proposed mechanism employs a two-fold overlay for each database that is to be included into the distributed SRS: a replication overlay that distributes databases to the sites that want to store a local copy of selected databases, and a query overlay that handles the actual queries.

5.1 Replication Overlay

The replication overlay is used to create copies of the databases used for SRS. It also performs database updates whenever the root database is modified.

This overlay reflects the actual usage pattern of SRS, where a certain site publishes biological data and other sites

Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03)
0--7695-1919-9/03 $17.00 © 2003 IEEE

rely on this data. It enables a site to be the authorative source for a certain compilation of biological data while allowing other sites to also store this data to speed up requests and become independent in case the authorative site becomes temporarily unavailable.

This replication overlay is to be created administratively upon installation of the distributed SRS: If a site's administrator wants to replicate a database, an overlay connection to another site with (a copy of) that database is created by listing that site in the configuration menu. Upon connection, the current state of the entire database is replicated. In the further course, each modification of the replication source also triggers an update of all the (connected) replicated sites.

Due to the tree-like character[1] of this overlay-network, both, scalability and timeliness of the replication process can be ensured. In practice, it can also be expected that the structure will reflect actual network topologies, since administrators are likely to configure the system to use nearby databases that provide sufficent bandwidth to enable timely updates.

5.2 Query Overlay

The query overlay routes requests to sites providing the requested database. Unlike the database replication overlay, this overlay is created automatically. Thus, in the typical end-user case, where an SRS installation is mainly used to access the distributed databases, administration effort is only required to add the locally provided databases that should be integrated.

For bootstrapping the query overlay, an SRS frontend needs to connect to a central directory server enlisting all the publicly available databases. Typically, for each database a small number of database replication sites is also listed for redundancy purposes. If a database is not publicly available, an additional authentification is required to retrieve the locations of this database. Although each such database provider may require authentification for the requests themselves, this mechanism hides private databases (and private replications) while giving all the front-ends operated in the respective private domain simple and full access to those databases.

After bootstrapping, no further centralized component will be required. This distributed approach renders the system both scalable and fault-tolerant. Instead, the query frontends will use the so-gathered lists to direct queries towards the respective providing sites for each database.

In the simplest case, each request message can and will be answered by one of the corresponding sites in that list. However, in order to benefit from the load-balancing and error-resilience properties of peer-to-peer overlay-networks, request-messages may be relayed to other nodes that are also capable and potentially more likely to answer a request. In such a case, a database-providing site does not respond to a query but replies with a relay message that indicates another database from the same replication tree.

Thereby, the front-end's database list collects more and more entries of database-providing sites being capable of responding to its requests. Additionally, each entry is accompanied by the observed latency with which that site responded to a request. If the number of entries for a certain database exceeds a certain limit, the entries with the most unfavorable latencies are dropped.

In order to separate the database handling from handling the request traffic, a site may relay a request to a relay node whose purpose is to redirect the request to either another relay node or a database that can handle the request. To that end, relay nodes maintain the same node lists as described above.

The latter creates a very special structure that goes beyond the plain structure of a replicated database or server cluster with a load-balancer, since this structure allows to form a very efficient, peer-to-peer-like, overlay-network.

This property is demonstrated in the following section.

6 Simulation Results

In order to validate the proposed mechanism, we have simulated a network of 500 installations of the distributed SRS with, on average, 10 concurrent users each. With that, we have measured both, response times as seen at the query front-ends and system load at the database access engines.

For the simulation we place the database access engines, relay nodes, and query front-ends randomly in a simple test network, a random graph embedded into a 2-dimensional Cartesian space, and assign the database access engines with random capacities, measured in concurrent users. Note, that we do not assume this number to be a hard limit, but rather a nominal load that optimally utilizes the available resources.

Figure 1 shows the correlation between the design capacity and the actual load that is achieved by our proposed mechanism. As can be seen easily, the mechanism succeeds in assigning each node almost exactly the number of concurrent users that it is able to handle. (The straight line depicts the optimal 100% load.)

Figure 2 shows the query latency experienced by the different SRS users. If the front-ends use arbitrary remote database access engines, our simulation yields latencies of around 55. The query relay mechanism described in section 5.2 brings this value down to around 10. This im-

[1] Due to manual interaction the system has to check if newly added connections comply with the tree structure.

Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03)
0--7695-1919-9/03 $17.00 © 2003 IEEE

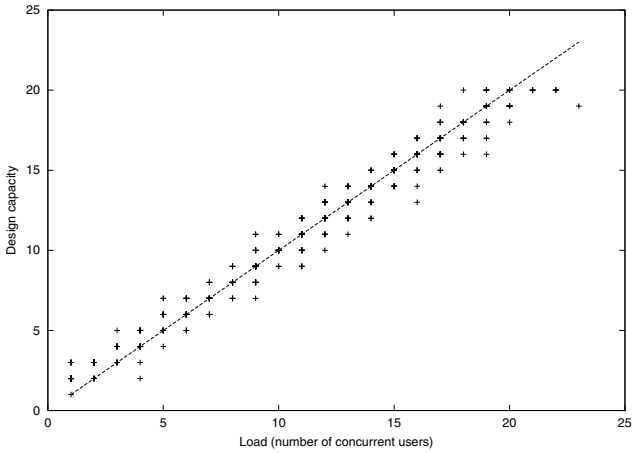

Figure 1. Correlation of design capabilities and actual load of the database access engines

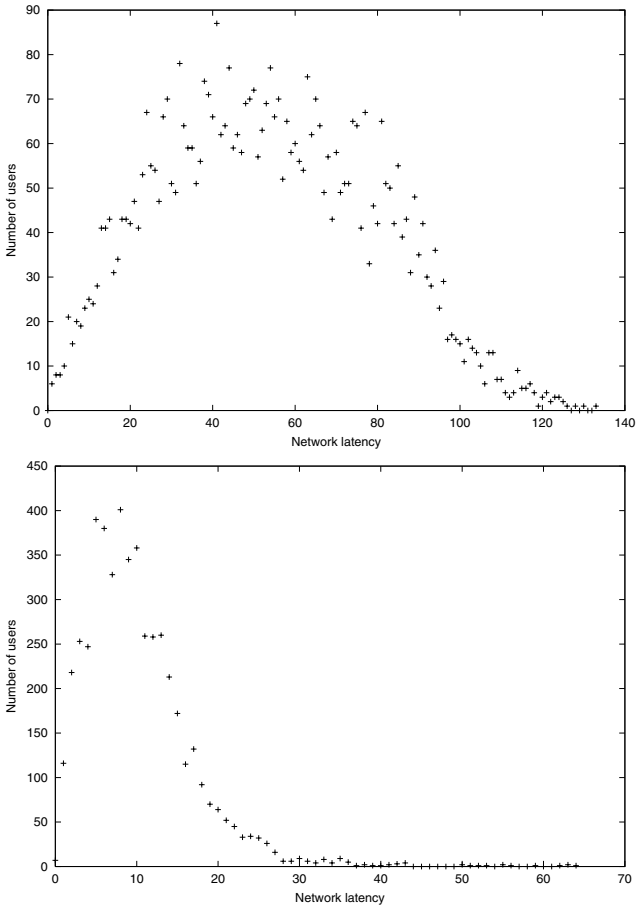

Figure 2. Latency experienced by the SRS users — Top figure shows random use of remote installations. Bottom figure shows improvement of the proposed mechanism

provement of more than a factor five compared to a straightforward overlay approach is due to the topology shaping properties incorporated in our overlay-formation mechanism.

Altogether, these results show that our proposed mechanism is able to achieve the design goals: It shapes a random peer-to-peer overlay-network in such a way that the network nodes are utilized according to their individual capacity. At the same time, the resulting network-topology minimizes latencies experienced by the queries.

7 Conclusions and Outlook

In this paper, we have described a mechanism that can turn the widely used SRS into a distributed system that benefits from (potentially all) the other SRS installations. Unlike with today's classic SRS, databases need not be replicated to every site that needs access to that database. Instead, all publicly available database installations can be used as if they were installed locally. These public databases can be supplemented by private databases not visible to and not accessible by the public.

This is achieved by establishing two different overlay-structures for each database. The first replicates the database to the sites that still want to store a local copy of that database. A second, much more important overlay-structure uses peer-to-peer overlay techniques to route requests to sites providing (a copy of) the respective database. Thereby, it guarantees resilience even in cases where several of the remote databases happen to fail. The special request routing mechanism described in this paper optimizes both, the request latencies experienced by the users and the load on the different database search engines. This is achieved by continuously tailoring the request overlay-network to the actually observed system load.

Compared to the classic SRS approach, this distributed mechanism does not only add more flexibility to the use of biochemical databases, it also increases the timeliness of the retrieved information and reduces the administration effort required to keep this data up-to-date. By employing the described overlay-mechanisms, the distributed SRS provides a reliable and responsive system that can benefit from potentially all the SRS installations world-wide.

Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03)
0--7695-1919-9/03 $17.00 © 2003 IEEE

References

[1] www.biodas.org/documents/spec.html.

[2] T. Etzold and P. Argos. SRS — an indexing and retrieval tool for flat file data libraries. *Comput Appl Biosci*, 9(1):49–57, Feb. 1993.

[3] T. Etzold and P. Argos. Transforming a set of biological flat file libraries to a fast access network. *Comput Appl Biosci*, 9(1):59–64, Feb. 1993.

[4] T. Etzold, A. Ulyanov, and P. Argos. SRS: information retrieval system for molecular biology data banks. *Methods Enzymol*, 266:114–128, 1996.

[5] D. P. Kreil and T. Etzold. DATABANKS — a catalogue database of molecular biology databases. *Trends in Biochemical Sciences*, pages 155–157, Apr. 1999.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proceedings of ACM SIGCOMM Conference*, pages 161–172, Aug. 2001.

[7] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware'01)*, pages 329–350, Heidelberg, Germany, Nov. 2001.

[8] E. M. Zdobnov, R. Lopez, R. Apweiler, and T. Etzold. The EBI SRS server — recent developments. *Bioinformatics*, 18(2):368–373, Feb. 2002.

[9] E. M. Zdobnov, R. Lopez, R. Apweiler, and T. Etzold. The EBI SRS server-new features. *Bioinformatics*, 18(8):1149–1150, Aug. 2002.

[10] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, Apr. 2001.

Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03)
0--7695-1919-9/03 $17.00 © 2003 IEEE