Preprint
UCRL-JC-150297

BioZoom: Exploiting Source-Capability Information for Integrated Access to Multiple Bioinformatics Data Sources

Ling Liu, David Buttler, Terence Critchlow, Wei Han, Henrique Paques, Calton Pu, and Dan Rocco

U.S. Department of Energy

Lawrence Livermore National Laboratory

This article was submitted to
Bioinformatics and Bioengineering 2003, Washington DC,
March 10–12, 2003

January 9, 2003

Approved for public release; further dissemination unlimited

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

This report has been reproduced directly from the best available copy.

Available electronically at http://www.doc.gov/bridge

Available for a processing fee to U.S. Department of Energy
And its contractors in paper from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-mail: reports@adonis.osti.gov

Available for the sale to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/ordering.htm

OR

Lawrence Livermore National Laboratory
Technical Information Department's Digital Library
http://www.llnl.gov/tid/Library.html

BioZoom: Exploiting Source-Capability Information for Integrated Access to Multiple Bioinformatics Data Sources

Ling Liu, David Buttler, Terence Critchlow[+], Wei Han, Henrique Paques, Calton Pu, Dan Rocco
Georgia Institute of Technology
College of Computing
email: {lingliu,buttler,weihan,paques,calton,rockdj}@cc.gatech.edu

[+]Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
critchlow@llnl.gov

Abstract. *Modern Bioinformatics data sources are widely used by molecular biologists for homology searching and new drug discovery. User-friendly and yet responsive access is one of the most desirable properties for integrated access to the rapidly growing, heterogeneous, and distributed collection of data sources. The increasing volume and diversity of digital information related to bioinformatics (such as genomes, protein sequences, protein structures, etc.) have led to a growing problem that conventional data management systems do not have, namely finding which information sources out of many candidate choices are the most relevant and most accessible to answer a given user query. We refer to this problem as the query routing problem. In this paper we introduce the notation and issues of query routing, and present a practical solution for designing a scalable query routing system based on multi-level progressive pruning strategies. The key idea is to create and maintain source-capability profiles independently, and to provide algorithms that can dynamically discover relevant information sources for a given query through the smart use of source profiles. Compared to the keyword-based indexing techniques adopted in most of the search engines and software, our approach offers fine-granularity of interest matching, thus it is more powerful and effective for handling queries with complex conditions.*

1 Introduction

Huge and growing amount of bioinformatics data that reside in specialized databases today, are accessible over the Internet, most of them with limited query processing capabilities. The Molecular Biology Database Collection [1, 4], for example, currently holds over 500 data sources, not even including many tools that analyze the information contained therein. The most popular resources including those concerned with protein sequences (such as SWISS-PROT, an annotated protein sequence database, and PIR, the Protein Information Resource), protein structure (such as PDB, the Protein Data Bank), genome data (such as AceDB, a *Caenorhabditis elegans* database), DNA (deoxyribonucleic acid) sequences (such as EMBL, the European Molecular Biology Laboratory and Gen Bank), motifs (such as PROSITE, a database of protein families and domains, and PRINTS, a compendium of protein fingerprints), and sequence matching (such as BLAST (Basic Local Alignment Search Tool) searches, available at several sites such as NCBI, EMBL, KEGG, DBJJ, and so forth).

It is widely recognized that Bioinformatics data sources are extremely helpful in assisting molecular biologists, geneticists, and biochemists to understand the biochemical function, chemical structure, and evolutionary history of organisms, and more importantly to use information collected or generated about human genome, such as protein sequences, DNA sequences, protein structure, chemical compounds, to design drugs to prevent and cure disease.

Bioinformatics data sources over the Internet have a wide range of query processing capabilities. Most Web-based sources allow only limited types of selection queries. Data from one source often must be combined with data from other sources to give scientists the information they need. Several data integration systems [5, 4, 11, 13, 3] have been created to provide users with integrated access and a single point of contact to multiple, heterogeneous bioinformatics data sources. One of the critical challenges for providing integrated access to bioinformatics data sources is the problem of effectively locating the right information from the right data sources and incorporating newly added capabilities or data sources in answering queries. More concretely, it is widely observed that not all the bioinformatics data sources can contribute to a query at any given time. Thus, it is important to route a query to only those data sources that are capable of answering the query. We refer to this problem as the query routing problem [8].

Query routing is a process of directing user queries to ap-

propriate servers by constraining the search space through query refinement and source selection. Concretely, effective query routing not only reduces the query response time and the overall processing cost, but also eliminates a lot of unnecessary communication overhead over the global networks and over the individual information sources.

Query routing is of particular importance to large-scale bioinformatics query systems for a number of reasons. First, popular online systems for searching life sciences data (such as genomic data sources) match queries to answers by comparing a query to each of the sequences in the data source. Efficiency in such exhaustive systems is crucial since some servers process over 40,000 queries per day [11]. Furthermore resolution of each query often requires comparison to over one gigabyte of genomic sequence data. While exhaustive systems are practical at present, they are becoming prohibitively expensive, even with database indexing techniques. Second, different bioinformatics data sources in differing formats have been set up to support different aspects of genomics, proteomics, and the drug design process. Some of these sources are huge and growing rapidly. Statistics show that bioinformatics data sources are now doubling in size every 15-16 months, and the number of users and the query rates are growing as well [6]. Third but not last, there are growing demands for answering simple key-word or string matching based queries with comprehensive categories of information. For instance, cancer researchers may expect to use an integrated bioinformatics query system to help identify genes that respond to low-doses of radiation. This problem is difficult because the information required by the scientists is spread across many independent, Web-based data sources, each using their own query interfaces with their own data formats and limited query processing capabilities. How to locate the relevant data sources that are capable of answering a query is critical to the performance of any integrated query system for transparent access to multiple bioinformatics data sources.

Surprisingly, most existing bioinformatics data integration systems [2, 3, 5, 10, 4, 13] do not provide the support for query routing even though some of them offer sophisticated query optimizations. Queries may be routed to data sources that are irrelevant or cannot contribute to the answers. As a result, not only is the response of queries delayed but also the throughput of the data servers is affected, not mentioning the additional network traffics incurred. In this paper we first present an overview of BioZoom and show how it can be used to integrate access to bioinformatics data from heterogeneous data sources. Then we introduce the BioZoom source-profile based query routing scheme, including the use of source-capability profiles to capture diverse and limited source content and query capabilities, and the multi-level progressive pruning algorithm for locating relevant data sources in answering queries from a large and growing collection of sources. To illuminate our discussion, we sketch several research scenarios that substantiate the need for cross-source queries and query routing based optimization. The main contribution of the paper is the concept of source-capability based query routing and its multi-level progressive pruning strategy for selecting the most relevant data sources in answering a bioinformatics query. We also report the first prototype development effort and our initial experimental result for the query routing algorithm.

2 BioZoom: An Overview

BioZoom is a bioinformatics data integration system that provides a single coherent framework for integrated access to a large, distributed collection of bioinformatics information providers. Before introducing the BioZoom query routing scheme, we first briefly overview the BioZoom system architecture and understand how the source-capability information is collected, and how BioZoom supports integrated access to multiple heterogeneous data sources. Figure 1 presents a sketch of BioZoom architecrure.

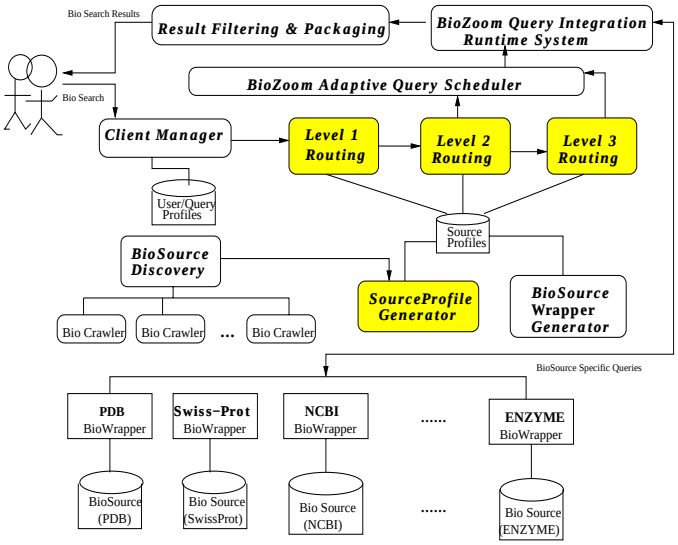

Figure 1: BioZoom System Architecture Sketch

Let us walk through the system architecture with an example of a simple genomic analysis task. To examine a DNA sequence in alignment with similar sequences from *blast* data sources, a scientist must use different tools with three different interfaces and convert the output from each one to a format acceptable to the next. More concretely, the scientist may start with a DNA sequence in a text file, then cut and paste the DNA sequence text into the search interface of a Blast data source, say *NCBI Blast*, to perform a search for similar sequences. The scientist would need to save results and extracts sequence identifiers of best matches manually and feed the sequence identifiers into another Web-based data source, say *PDB Blast*, to retrieve full-length sequence text of best matches. Again the scientist needs to save re-

sults and converts format in order to use a command-line tool to create a multiple sequence alignment [13].

Using BioZoom, the scientist first needs to enter the DNA sequence as the search keyword and select the output options such as generate a multiple sequence alignment using the best matches of similar sequences. Once the client manager parses the query. The level-one query routing will identify the types of candidate data sources needed to answer this query. The level-two routing will prune the set of candidate data sources based on their query capabilities. The adaptive query scheduler generates corresponding subqueries to data sources selected by the first two levels of routing, and defines the ordering (schedule) of executing subqueries. The level-three routing collects the types of dynamic information needed for relevance reasoning and perform further irrelevance pruning at runtime. The results returned from selected data sources by the BioZoom runtime system will feed into the result filtering and packaging module to perform the final stage of query fusion. The fused query results will be returned to the scientist on the screen or delivered in a file. The source capability profile generator is used to infer the source capability information from the source profiles created by the Bio Crawlers, which are Web robots that traverse the Web and look for interesting bioinformatics data sources and extract their query capabilities.

3 Capability Based Routing

We begin by introducing a motivating example and a predicate metadata model in which user queries, user query profiles, and source profiles are captured. Then we present the design of our query routing algorithms.

3.1 The Motivating Example

Bioinformatics sources available over the Internet have diverse and yet limited query processing capabilities. Most information servers, where data actually reside (such as PDB, NCBI, EMBL), only support limited types of selection or similarity queries. This introduces some interesting query processing challenges as illustrated below.

Example 1. Consider a pharmaceutical researcher who wants to research drugs to combat HIV. To understand the approach the researcher may take to combat this virus, it is important to understand how the virus works.

The HIV virus itself is composed of two RNA strands encased in a protein envelope. The viral envelope has 2 proteins, named gp120 and gp41. The gp120 binds to CD4, a receptor protein on a type of white blood cell, called CD4+ T cells. The gp41 then causes the fusion of the HIV with the T cell. After the virus has merged with a cell, the viral RNA is inserted into the cytoplasm of the cell. Each virus particle has 2 copies of an RNA genome, which are transcribed into DNA in the infected cell and integrated into the host cell chromosome with the help of an enzyme called reverse transcriptase. The viral RNA copies itself into the DNA of the cell, causing the cell to produce more of the viral RNA. The RNA transcripts produced from the integrated viral DNA serve both as mRNA to direct the synthesis of the viral proteins and later as the RNA genomes of new viral particles, which escape from the cell by budding from the plasma membrane, each in its own membrane envelope. For more details on how HIV operates, see (http://www.niaid.nih.gov/factsheets/howhiv.htm).

One possible solution is to use drugs to prevent the virus from attaching to receptors on cells so that other white blood cells, called killer T cells, can recognize, ingest, and destroy the viral package before it has a chance to infect a new cell. There are many ways of developing such drugs. One common method is experimental; the process is to physically test a compound against a sample of the virus or of a protein that the virus binds to. This process may be labor intensive. In addition, choosing a compound to test is quite difficult because most pharmaceutical companies have a catalogue of several million compounds making an exhaustive search extremely slow, tedious, and error-prone. Another way to develop drugs is search properties of known compounds for promising candidates. There are several techniques that are relevant here. First, a researcher may find all related proteins to a protein that is known to be involved in a disease process, such as gp120, or CD4. Second, a researcher may want to find all drug compounds that are similar to a drug that affects the protein they are interested in (for example, a reverse transcriptase inhibitor or a drug that inhibits the binding of gp120 to CD4). A third technique is to analytically determine the effect of a drug compound on the protein and the related proteins. Some efforts have been made in modeling how chemical compounds affect a protein based on a computational model of the chemical and protein as well as with comparisons with known interactions between the protein and a similar chemical, or between the chemical and a similar protein.

Consider the example query. First, a researcher may use PDB to find the structure for CD4, gp120, or gp41. Then the researcher may wish to find similar proteins to compare structure, function, or related research. To do this, he needs to take the sequence encoding of the proteins discussed above, such as gp120, translate the sequence given by PDB into a sequence suitable for searching other data sources. Such a translation is often done by replacing amino acid names with their single letter encoding. Then he takes this sequence and submits it to multiple similarity matching sources. Examples include NCBI's BLAST tool − blastp, or one of its many mirror sites and other BLAST sites that are not strict mirrors of NCBI, such as EMBL, DBJJ, or KEGG. The blast searches will list proteins similar to the one submitted, as well as how the amino acid sequence aligns with each of the similar proteins. Finally, all of the relevant publications, in a literature databases such as PubMed, for the similar proteins will be gathered. Now the researcher enters the drug design step. First, he needs to understand

how chemical compounds can affect the proteins identified. The goal here is to find a chemical that will inhibit gp120 from binding with the CD4 receptor, while minimizing the interference with regular cellular function.

Pharmaceutical companies usually maintain a list of chemical compounds with on the order of a million entries. Through mathematical modeling, how each compound interacts with the physical structure of each protein identified with the function of HIV can be predicted (with varying degrees of success). Studies of how the chemical has affected the function of similar proteins is another way of predicting how the chemical will interact of a protein. We can express a query searching for the proteins identified above as well as all similar proteins as $similar(1, \{\ keyword = \text{"HIV"} \wedge (protein=\text{"gp120"} \vee protein=\text{"gp41"} \vee protein=\text{"CD4"}\})$. This search is fairly complex and cannot be processed by any known source in one-stop. Thus, to process this query, our system needs to break down the end-user query into source-specific queries that are executable at individual sites, such as NCBI, PDB, or EMBL. One possible plan is to break the query into the following series of queries. (1) Query PDB: $(keyword = \text{"HIV"} \wedge (protein=\text{"gp120"} \vee protein=\text{"gp41"} \vee protein=\text{"CD4"})$, obtaining the structure of these known proteins; (2) For any protein r from the result of (1), convert r into a protein sequence r_s; (3) at NCBI execute a BLAST query for each r_s; (4) filter out all results that are not with-in a similarity of 1, as defined by the BLAST similarity measure.

Through this example, we observe two interesting facts. First, the extraction and use of the PDB and NCBI source profiles plays a critical role in routing the query to these relevant data sources. Second, even simple selection queries against a single data source across the Internet may have more complications due to the source-specific content and its limited query capability. The situation becomes more sophisticated when we have queries over multiple distributed data sources that are heterogeneous in both information content and their query capabilities.

3.2 The Metadata Description Model

The metadata description model [8, 7] is designed to be an object relational model. Typical components of the metadata model are classes, a set of (simple or composite) attributes associated with each class, a class hierarchy described by a subclass-superclass partial order. We use a unary relation to describe each class and a binary relation to describe each attribute.

We model queries with select, project, join, and union operations and the built-in comparison predicates such as $\leq, <, =$ and $\neq$. We assume set semantics for queries. For convenience of our analysis, we consider only *conjunctive queries*. A conjunctive query Q consists of a head predicate with arguments, denoting the result template, and a body, representing a binding pattern [12] of Q. The arguments of the predicate that are provided as input parameters of the query are expected to be bound. The arguments of the predicate that are produced as outputs of the query are free variables. We use lower case letters for variable names and uppercase letters with bars to denote tuples of variables and constants. We describe a conjunctive query Q by a quadruple $(Q_{from}, Q_{in}, Q_{out}, Q_{cond})$ where Q_{from} is the set of virtual types used in Q, Q_{in} is the set of input arguments, Q_{out} is the set of output arguments, and Q_{cond} is the conjunction of comparison atoms.

A user may pose queries on the fly (without using any predefined views or classes). For each user query and the result patterns, we create a set of *virtual object types* as its result place holder, which describes all the arguments used in the query, including the classes or relations, the data types, the domain constraints, and the usage (i.e., as input or output parameter) of the arguments.

Example 2. Consider online BLAST search sites (sources) for protein sequence similarity, such as NCBI. Suppose we want to search for a similar protein sequence, protein structure, and related research to the protein *gp120*, published in 2001 to better gauge the effects of a new drug. We may express the query Q: *find similar protein sequence, protein structure, and related research, for* gp120 *where publication year = 2001* as a conjunctive query of the following form:

$$\begin{aligned} query(sq, st, a, t, j) :- \ & Protein(p),\ Literature(m), \\ & sequence(p, sq),\ structure(p, st),\ author(m, a), \\ & year(m, y),\ title(m, s),\ journal(m, t), \\ & y = 2001 \wedge \textsc{Similar}(p, \text{'}gp120\text{'}). \end{aligned}$$

$query(sq, st, a, t, j)$ is the *head* of the query, and its arguments protein sequence p, structure s, author a, title t, journal j are its *distinguished variables*. In terms of relational SQL, the distinguished variables of the query correspond to attributes appearing in the `SELECT` clause. The rest are *atoms* of the *body* of the query, and are the bounding pattern of the query. Note that the equality predicates in the `WHERE` clause are represented by equating variables in different atoms of a conjunctive query. The following is the internal representation of this conjunctive query:

$$\begin{aligned} Q_{from} &= \{Protein(p), Literature(m)\}, \\ Q_{in} &= \{sequence(p, \text{'gp120'}),\ year(m, 2001)\}, \\ Q_{out} &= \{sequence(p, sq),\ structure(p, st), author(m, a), \\ & \quad title(m, t), journal(m, j)\}, \\ Q_{cond} &= \{BLAST(sq, \text{'gp120'}),\ y = 2001\} \end{aligned}$$

The researcher who poses the query does not need to be aware of what information sources are currently available and which data schemas or pre-defined views should be used to access them. The data independence as such allows the query routing to incorporate newly added information sources seamlessly into the system without affecting the way how queries are posed and how answers are delivered, thus higher scalability is achieved, especially when the collection of information sources available is large and frequently changing.

Before we show how the query is routed to the most relevant data sources, we first introduce the concept of source-capability profiles, which play a critical role in pruning irrelevant data sources.

3.3 The Source Capability Profile

A source capability profile tells what is in an information source (content description) and what types of services (capability description) are provided about its content. It contains not only the content and query capability description but also statistics on the local data (e.g., size of relations), availability of the source with respect to the access cost and access authorization, as well as update frequency and capabilities of the source. In addition, each source may export information about itself by giving values to a list of meta attributes such as `FieldSupported` (list of optional fields), `Linkage`(the URL where the source should be queried), `ContentSummaryLinkage` (the URL of the content summary of the source). In this section we will focus only on the source category, content, and query capability descriptions, since they are the essential components of the source profile and are used extensively in each step of the query routing process.

The *category and content* description of an information source describes what is in the information source. The content description of an information source tells us what types of objects are in the source. The category description tells us what type of domain the source data are used for and the *IsA* categorization of the source. The source category description often contains information that can be used to verify an input (selection) condition or fill in an output parameter of a query. We model the contents of an information source in terms of the object types and the object access constraints that the source objects must satisfy. Each source object type is described by a unary relation. Each access constraint is described using a conjunction of built-in comparison atoms of the form $a\theta v$ where a is an attribute of a source type and v is a constant drawn from a domain that is compatible to the domain of a. We may view a source content description as a collection of views defined over the source.

The *query capability* description of an information source tells which types of queries the source can answer about its content. We model the query capabilities of an information source S using *capability records*, each is denoted by $(S_{in}, S_{out}, S_{cond})$. S_{in} denotes the set of permissible input arguments. S_{out} denotes the set of permissible output arguments. S_{cond} denotes the logical constraint ($\wedge$ or $\vee$) on the mandatory input arguments.

In summary, we denote each information source by a triplet $(S_{cat}, S_{cnt}, S_{qpd})$ where S_{cat} denotes the text description of the category of the source, S_{cnt} is a set of source relations, each may be associated with some access constraints. S_{qpd} denotes a set of query capability descriptions, each is of the form $(S_{in}, S_{out}, S_{cond})$ (see Figure 4).

Consider the query in Example 2. Suppose we have extracted and collected the source profiles of the information servers as shown in Figure 4, among many others. Using the user query profile and the source profiles, we may conclude, without running the query, that some of the sources are obviously not contributing to the answer of Q. For instance, we can immediately determine that Sources 7, 8, and 9 are not relevant to this query, because they are focused on Motif searches and not protein sequence similarity. We can also conclude that Sources 4, 6, 10 are not able to contribute to the answer of Q. However, the reasoning here is more subtle. We are interested only in articles that are published in 2001. However, Source 6 only has articles published before 1965. Source 4 requires a file input format that is not available from the known output of any of the sources, and Source 10 only provies book information. Thus, we are left with sources 1, 2, 3, and 5. The mechanism used in making such routing decision will be described in Section 3.4. Readers may refer to [7, 8] for further detail on how to obtain source profile and query routing techniques.

3.4 Query Routing: The Main Steps

The ultimate goal of query routing is to constrain the search space for a query over a large collection of available information sources by reducing the overhead of contacting the information sources that do not contribute to the query answer.

Given a user query Q, a user query profile of Q, and a set of source content and capability descriptions, we design the query routing service as a two-phase process. At the query refinement phase, mechanisms are applied to refine the original query into a well focused query, aiming at reducing the false positives in the query result set and enhancing the quality and the degree of accuracy of the results produced from source selection. In this section, we concentrate on the second-phase of the query routing task – source selection and its two-step routing process. Readers may refer to [7] for the detailed algorithms and further discussions.

Step 1: Level-one relevance pruning.
This step serves as the first-round selection which discovers the candidate information sources whose content descriptions are in some ways related to the scope of a query Q (e.g., in terms of substring matching or in concept similarity). Other factors such as unavailability of the sources or affordability of the sources should be considered at this step too. For level-one relevance pruning we use the user query scope description of Q and the content and category description of the sources. Source profiles that are redundant (covering the same information from the same source) are also removed at this stage. We call the set of sources selected by this step as *target* information sources of *level-one relevance*.

Consider Example 2 query, level-one relevance pruning will prune the information sources whose contents are not relevant to biomedical literature, protein sequence structure,

or protein sequence similarity, based on the list of source profile descriptions in Figure 4, among many others. It will find that sources 7, 8, and 9 are not relevant to the query answer.

Step 2: Level-two relevance pruning.
This step prunes the information sources that have level-one relevance but do not offer enough query capability to contribute to the answer of Q. The decision is made based on the input and output arguments of Q, the user query profile of Q, and the query capability descriptions of the sources. The user query capacity description of Q and the source profiles are used in the level-two relevance pruning. We call the set of sources selected by Step 2 as *target* information sources of *level two relevance*.

The process for level-two relevance pruning has two phases. In the first phase we prune the information sources of the following three cases:

- (1) data sources that have no input or output arguments, which are relevant to the arguments used in the user query, or
- (2) data sources that have conflict with the interest of the user query (such as the query selection conditions do not match the access constraints of the sources), or
- (3) data sources that have arguments corresponding to the mandatory input parameters of the user query but these arguments can only be used as input and are not included in the list of output arguments of the sources.

The information sources selected in the first phase will be passed to the second phase where more sophisticated pruning is conducted in the process of generating an executable plan of the query. For example, the following two additional cases are pruned accordingly:

- (4) date sources (say S_i) whose output capability are not enough to satisfy the input requirement of the other sources (say S_j) when an inter-site join from S_i to S_j is required.
- (5) Date sources whose mandatory input requirement is higher than the input arguments that the user query provides, and there are no other information sources executed earlier which would have enough output capability to complement such requirement.

Consider Example 2 query, level-two relevance pruning will further prune away Sources 4, 6, and 10 because they are incapable of contributing to the query answer due to the restriction on the scope of query interest (e.g., Source 10 is a bookstore not an technical article source or protein database), or the constraints on the list of mandatory input or output arguments of the sources (e.g., Source 4 requires a file input), or the conflict of query interest with the access constraints associated with the sources (e.g., Source 6 only provides citations for articles published before 1965, whereas the query is interested in only articles published in 2001).

In addition, there is a need to identify and prune mirrored or replicated sources. This step is usually delayed until last to allow for the greatest flexibility in choosing the most effective data source for a particular query.

A prototype of the BioZoom query routing subsystem is currently under testing. Figure 2 shows an example run of the first and second level routing, the query scheduling, and query execution in the first prototype of BioZoom. The source capability profiles used in this version is generated manually. We are working on building the first version of the Bio Crawlers by extending XWRAP Elite toolkit [9]. The source capability profiles are inferred based on the source information collected by the Bio crawlers.

3.5 Initial Experiments

We report two experimental results in this paper. All routing measurements were taken on a Sun E450 server, with 4 400-MHz UltraSPARC-II processors, and 1 GB of RAM running Solaris 7. The software was implemented in Java and run on the Java HotSpot Client virtual machine (build 1.3.0, mixed mode). Each experiment was run 20 times and the results were averaged to mitigate the impact of start-up costs such as just-in-time compilation of methods, and other execution anomalies such as garbage collection.

The essence of routing is to determine the relevance of a source to a particular query. There are two typical ways in which a data source is measured for relevance: content relevance and source capability relevance. Content relevance is often determined by keyword relevance ranking. Source-capability relevance selects sources that have the capability to answer the queries.

In the first experiment, we vary the percentage of sources that are categorically relevant and completely relevant at the schema level. The first figure on the left side of Figure 3 shows the execution time required for querying between 20-1000 sources, where 30% to 40% of sources are irrelevant and ten threads are used to retrieve data in parallel. Obviously, query fusion for integrated access without routing performs the worst. Query fusion with level two routing performs better than query fusion with only level-one routing. Note that in our routing scheme, level two routing is built on top of level one routing results.

In the second experiment shown on the right side of Figure 3, it demonstrates that arbitrarily increasing the number of threads servicing a single query suffers from the law of diminishing returns. Using 20 threads rather than 10, execution time without routing is nearly halved. However, using 70 threads is only 14% faster than using 60 threads. This is even more pronounced when routing is used, because when the number of selected data sources is below 60, adding additional threads will not speed up the process.

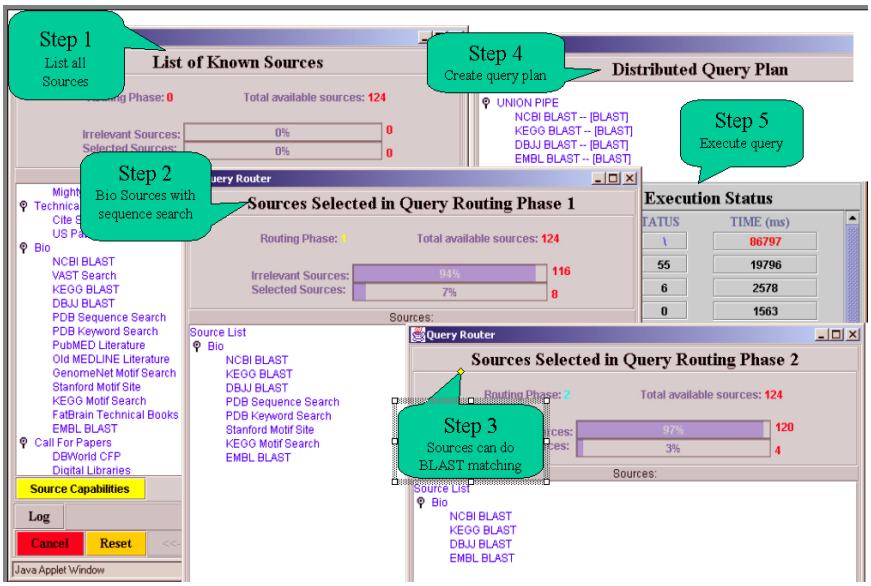

Figure 2: An Example Run of Query Routing and Execution in BioZoom

4 Related Work and Conclusion

The very nature of scientific research and discovery leads to the continuous creation of information that is new in content or representation or both. Despite the efforts to fit molecular biology information into standard formats and repositories such as the PDB (Protein Data Bank) and NCBI, the number of databases and their content have been growing, pushing the envelope of standardization efforts such as mmCIF [14]. Providing integrated and uniform access to these databases has been a serious research challenge. Several efforts [2, 5, 4, 11, 13, 3] have sought to alleviate the interoperability issue, by translating queries from a uniform query language into the native query capabilities supported by the individual data sources. Typically, these previous efforts address the interoperability problem from a digital library point of view, i.e., they treat individual databases as well-known sources of existing information. While they provide a valuable service, due to the growing rate of scientific discovery, an increasing amount of new information (the kind of hot-off-the-bench information that scientists would be most interested in) falls outside the capability of these previous interoperability systems or services.

In this paper, we address the problem of providing automated or semi-automated access to new information that has just become available, sometimes by changing the representation format of an existing database. The lag between the discovery and making the information available is primarily due to the human intervention needed to translate the new information either to an existing database format, or to augment a database with new formats or fields. We believe that the increasing rate of scientific discovery and publication will make this problem increasingly serious, since more databases will be augmented more frequently, or new databases will be created to publish the new information.

We have described the BioZoom query routing scheme for providing fast access to the growing new information that remains elusive with the current technology. The main contribution of the paper is the application of the concepts and techniques called *query routing* to increase the degree of automation in new information access and to reduce the amount of unnecessary delay due to contacting sources that cannot contribute to given queries. Query routing uses metadata, called *source-capability profile*, to support dynamic matching of each query with the information sources that are relevant to, and capable of, responding to that query. By updating a source profile, new information or capabilities of the source are immediately accessible by queries processed through query routing. Furthermore, addition of new bioinformatics data sources and capabilities can be dynamically incorporated into the subsequent execution of queries. The main thrust of our query routing scheme is its intelligent source selection powered by the source-profile based multi-level progressive pruning strategy. Our initial experiments show the increased benefits of query routing as the number of data sources available to a query increases.

Acknowledgement. This work was partially conducted under the DoE SciDAC project, LLNL Large-scale data access (LDRD) project, a grant from NSF CCR and a DARPA PECS project.

This work was performed for DOE by UC/LLNL contract no. W-7405-Eng-48.

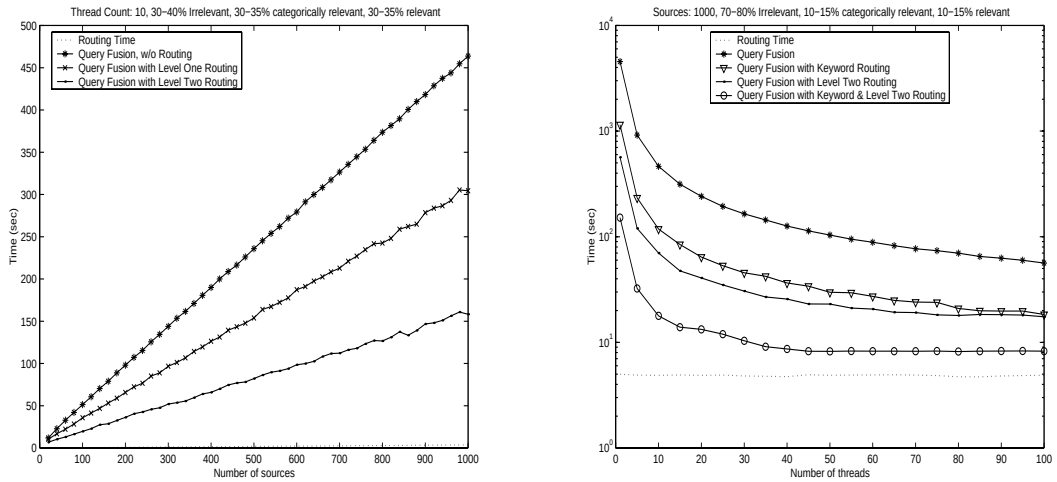

Figure 3: Performance Gains From Routing

References

[1] A. Baxevanis. The molecular biology database collection: An online compilation of relevant database resources. *Nucleic Acids Research*, 28(1):1–7, 2000.

[2] T. Critchlow, K. Fidelis, M. Ganesh, R. Musick, and T. Slezak. Datafoundry: Information management for scientific data. *IEEE Transactions on Information Technology in Biomedicine*, 4(1):52–57, March 2000.

[3] S. Davidson, O. Buneman, J. Crabtree, V. Tannen, G. Overton, and L. Wong. Biokleisli: Integrating biomedical data and analysis packages. *Bioinformatics: Databases and Systems*, S. Letovsky, Editor, Kluwer Academic Publishers, Norwell, MA:201–211, 1999.

[4] C. A. Goble, R. Stevens, G. Ng, S. Bechhofer, N. W. Paton, P. G. Baker, M. Peim, and A. Brass. Transparent access to multiple bioinformatics information sources. *IBM Systems Journal*, 40(2):532–551, 2001.

[5] L. Haas, P. Schwarz, P. Kodali, E. Kotlar, J. Rice, and W. Swope. Discoverylink: A system for integrated access to life sciences data sources. *IBM Systems Journal*, 40(2):489–511, 2001.

[6] N. Huyn. Data analysis and mining in the life sciences. *ACM SIGMOD Record*, 30(3), 2001.

[7] L. Liu. Query routing: An intelligent query service for accessing heterogeneous information sources. Technical report, TR97-10, Department of Computing Science, University of Alberta, 1997.

[8] L. Liu. Query routing in large-scale digital library systems. In *Proceedings of the 15th International Conference on Data Engineering (ICDE)*, IEEE Press, March 1999.

[9] L. Liu, C. Pu, and W. Han. XWrap: An XML-enabled Wrapper Construction System for Web Information Sources. *Proceedings of the International Conference on Data Engineering*, pages 611–621, 2000.

[10] A. Markowitz, I. Chen, A. Kosky, and E. Szeto. Opm: Object-protocol model data management tools'97. *Bioinformatics: Databases and Systems*, S. Letovsky, Editor, Kluwer Academic Publishers, Norwell, MA:187–199, 1999.

[11] S. McGinnis. (genbank user services, national center for biotechnology information (ncbi), national library of medicine, us national institute of health). *Personal Communication*, 1Jan., 1998.

[12] A. Rajarman, Y. Sagiv, and H. Ullman. Answering queries using templates with binding patterns. In *Proceedings of PODS*, 1995.

[13] A. C. Siepel, A. N. Tolopko, A. D. Farmer, P. A. Steadman, F. D. Schilkey, B. Perry, and W. D. Beavis. An integration platform for heterogeneous bioinformatics software components. *IBM Systems Journal*, 40(2):570–591, 2001.

[14] J. Westbrook and P. Bourne. Star/mmcif: An extensive ontology for macromolecular structure and beyond. *Bioinformatics*, 16(2):159–168, 2000.

Source 1: NCBI BLAST
Category: *Protein and nucleotide similarity search;* **Content**: *sequence(s), database = {Drosophila, est_human, ... }*
Query Capabilities: $\{protein_sequence(s,p), nucleotide_sequence(s,n), database(s,db), searchtype(s,t)\}$,
$\{similar_sequence(s,sim), alignment(s,a), sequence_id(s,id)\}$,
$\{(protein_sequence(s,p) \wedge database \wedge searchtype = (blastp \vee blastx \vee tblastx)) \vee$
$(nucleotide_sequence(s,p) \wedge database \wedge searchtype = (blastn \vee tblastn))\}$.
Mirrors:*Austrailia, China, Japan, Korea, Malaysia, Singapore, Thailand, USA*

Source 2: KEGG
Category: *Protein similarity search;* **Content**: *sequence(s), database = { Drosophila melanogaster, Homo sapiens, ... }*
Query Capabilities: $\{protein_sequence(s,p), nucleotide_sequence(s,n), database(s,db), searchtype(s,t)\}$,
$\{similar_sequence(s,sim), alignment(s,a), sequence_id(s,id)\}$
$\{(protein_sequence(s,p) \wedge database \wedge searchtype = blastp) \vee$
$(nucleotide_sequence(s,p) \wedge database \wedge searchtype = blastn)\}$.

Source 3: PDB
Category: *Protein structure search;* **Content**: *structure(t) {All known protein structures}*
Query Capabilities: $\{protein_sequence(t,sq), pdb_id(t,id), title(t,tt)\}$,
$\{pdb_id(t,id), title(t,tt), sequence(t,sq), classification(t,c), compound(t,cmpnd), structure(t,st)\}$,
$\{protein_sequence(t,sq) \vee pdb_id(t,id) \vee title(t,tt)\}$
Mirrors: *San Diego Supercomputer Center; Rutgers University; National Institute of Standards and Technology; Cambridge Crystallographic Data Centre, UK; National University of Singapore; Osaka University, Japan; Universidade Federal de Minas Gerais, Brazil; Bio Molecular Engineering Research Center, Boston University Brookhaven National Laboratory ...*

Source 4: VAST
Category: *Protein structure similarity search;* **Content**: *structure(t), database = (non-redundant PDB, or full PDB)*
Query Capabilities: $\{PDB_file(t,pdb_f), XPLOR_PDB_file(t,xplor_pdf_f), CNS_deposit_file(t,cns_deposit_f), \}$,
$\{sequence(t,sq), structure(t,st)\}$,
$\{PDB_file \vee XPLOR_PDB_file \vee CNS_deposit_file\}$

Source 5: PubMed
Category: *Biomedical literature;* **Content**: *Article(a)*
Query Capabilities: $\{author(a,u), title(a,t), keyword(a,k), pubmed_id(a,id)\}$,
$\{author(a,u), title(a,t), journal(a,j), pub_year(a,y), volume(a,v), issue(a,i), pages(a,p), pubmed_id(a,id)\}$,
$\{author(a,k) \vee title(a,t) \vee keyword(a,k)\}$.

Source 6: Old MEDLINE
Category: *Biomedical literature;* **Content**: *Article(a), pub_year(a,y)* $\wedge y < 1965$
Query Capabilities: $\{author(a,u), title(a,t), keyword(a,k), pubmed_id(a,id)\}$,
$\{author(a,u), title(a,t), journal(a,j), pub_year(a,y), volume(a,v), issue(a,i), pages(a,p), pubmed_id(a,id)\}$,
$\{author(a,k) \vee title(a,t) \vee keyword(a,k)\}$.

Source 7: GenomeNet (Japan)
Category: *Motif Search* **Content**: *Motif(f); database = (Vertebrates, Virus, Insects, Plants, Bacteria,Fungi, Nematodes), search_type = (Prosite Pattern, Prosite Profile, BLOCKS, ProDom, PRINTS, Pfam)*
Query Capabilities: $\{sequence(f,s), database(f,db)\}$,
$\{expectation(f,e), probability(f,p), description(f,d)\}$,
$\{sequence(f,s) \wedge database(f,db) \wedge search_type(f,srcht)\}$.

Source 8: Stanford Motif search
Category: *Motif Search* **Content**: *Motif(f); database = (BLOCKS+ and PRINTS , non-biased BLOCKS+ and PRINTS)*
Query Capabilities: $\{sequence(f,s), database(f,db)\}$,
$\{expectation(f,e), probability(f,p), description(f,d)\}$,
$\{sequence(f,s) \wedge database(f,db)\}$.

Source 9: KEGG Motif
Category: *Motif Search* **Content**: *Motif(f); database = (D.melanogaster, E.coli, H.sapiens ...), Prosite patterns*
Query Capabilities: $\{sequence(f,s), database(f,db)\}$,
$\{expectation(f,e), probability(f,p), description(f,d)\}$,
$\{sequence(f,s) \wedge database(f,db)\}$.

Source 10: fatbrain.com Book Store Database
Category: *Technical Book Store (includuing biotech books);* **Content**: *Book(b);*
Query Capabilities: $\{title(b,t), authors(b,a)\}$,
$\{title(b,t), authors(b,a), publisher(b,pub), year(b,y), price(b,p), isbn(b,n)\}$,
$\{title(b,t) \vee authors(b,a) \vee isbn(b,n)\}$.

Figure 4: A sketch of source capability description of example data sources