Large-Scale Collaborative Analysis and Extraction of Web Data

Felix Weigel Biswanath Panda Mirek Riedewald
Johannes Gehrke Manuel Calimlim
Dept. of Computer Science, Cornell University
{weigel,bpanda,mirek,johannes,calimlim}@cs.cornell.edu

ABSTRACT

Archived web data is a great resource for scientific research, but poses serious challenges in data processing and management. We demonstrate the *Web Lab Collaboration Server*, a platform and service for large-scale collaborative web data analysis in a distributed computing environment, and show how it seamlessly supports non-technical users during search, data extraction and analysis.

1. INTRODUCTION

The web has evolved into a large and rich data source for researchers in many different disciplines like social science, economics, linguistics, business and marketing. For example, social scientists study the structure and dynamics of social networks in on-line communities and the blogosphere; economists search for trends in stock markets; marketing specialists analyze customer preferences and feedback; and linguists track the spread of neologisms.

The *Internet Archive* [3] and other public or commercial archiving services make web crawls persistently available for analysis. For example, the Internet Archive has collected petabytes of crawled web data since 1996 which is accessible to researchers. However, even with such a wealth of data publicly available, there are three major obstacles in creating effective and practical analysis applications:

Extraction of structured data sets: Researchers are typically interested in subsets of the data with specific contents and structure. For example, a social scientist studying a social network first has to extract the network graph from a set of archived web pages selected from a specific site or window of time. Preparing such customized data sets requires writing extraction scripts tailored to the task at hand. This is not only time-consuming, but also hard to accomplish for many researchers in disciplines outside computer science.

Cleaning and formatting data sets: Web data often contains errors, omissions, inconsistencies and outdated information that must be corrected before the analysis. Moreover, extracted data may need to be reformatted to serve as input to existing analysis tools. These steps are often needlessly repeated by different users.

Efficiency and scalability issues during analysis: Processing gigabytes or even terabytes of data can easily require more computing resources than are available at a researcher's desktop or lab. But even with sufficient hardware, it is non-trivial to write analysis code that takes advantage of parallelism, shared memory or distributed computing power and storage, features that are key to large-scale data analysis.

As a first step toward addressing the challenges in an end-to-end solution, we present the *Web Lab Collaboration Server*, a platform and service for large scale analysis of web data. The system is designed and built on three key observations:

High-level interface: Many users interested in analyzing web data are experts in domains outside computer science. Such users can benefit from an intuitive interface that hides technical details about how the different components work and interact. There is a significant amount of work on graphical web data extraction using wrappers [5], and on large-scale distributed data analysis using the Map/Reduce paradigm [1]. More recently, languages like Pig Latin [6], Sawzall [7] and Dryad [4] have been proposed to facilitate writing Map/Reduce tasks. However, implementing a complete extraction and analysis workflow using these techniques is cumbersome for non-technical users. In this demonstration, we show that complex extraction and analysis tasks can be accomplished through a simple intuitive GUI, while building on established Map/Reduce technology in the backend.

Collaborative infrastructure: Data extraction, cleaning and formatting are time-consuming and tedious tasks. Once data sets have been prepared for analysis, they should be managed in a central repository to enable reuse and sharing among a community of researchers. The same applies to the analysis protocols and results and the extraction code contributed by different users.

Software-as-a-Service approach: Setting up and maintaining a large data repository and computing infrastructure is a complex endeavor, especially as more data becomes available and analysis requirements evolve. This complexity should be hidden from the users. However, while considerable effort has gone into data collection and parallel processing, the problem of convenient access for ad-hoc analysis has been largely ignored. We describe a web-based architecture that enables users to leverage a powerful distributed computing and archiving platform for their extraction and analysis tasks, and demonstrate two applications that use our infrastructure

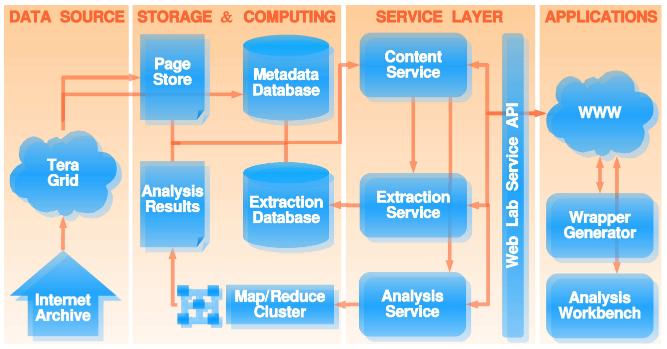


Figure 1: System architecture of the Web Lab Collaboration Server.

to process web crawls from the Internet Archive [3] through remote services. We are not aware of any other system providing users with such an end-to-end solution to large-scale data extraction and analysis.

The rest of the paper is organized as follows. Section 2 describes the system architecture. Section 3 walks through an example analysis task to demonstrate the current data extraction and analysis functionality of the Web Lab Collaboration Server. Section 4 summarizes our contributions.

2. SYSTEM OVERVIEW

Figure 1 depicts the system architecture of the Web Lab Collaboration Server. The core components are the storage and computing infrastructure and the service layer (center). These components are hosted at Cornell University, separated physically from both the crawler (left) and the client applications (right). We periodically transfer web crawls from the Internet Archive [3] in San Francisco through a high-speed TeraGrid connection and store them in our *Data Repository*. To date, we have downloaded more than 30 TB of compressed web data this way. In addition to the full-text content of the web pages, which is stored in compressed files, we keep metadata about each page in a relational database. This metadata comprises the URLs, crawl times, mime types, outlinks, anchor texts etc. of all pages, plus pointers to the corresponding full-text files on disk.

A *Content Service* provides clients with access to the full text and metadata of archived pages and to extracted data sets and analysis results. For example, our project web site [9] features a search tool that internally uses the Content Service to retrieve crawled pages based on URL patterns, time stamps, etc. The Content Service also serves the data extraction and analysis applications described below. Together, they are part of a *Web Lab Service API* that lets clients access the Data Repository through JSON RPC and Java APIs. The API is still being expanded and includes more general-purpose services that are not shown in Figure 1, e.g., for progress monitoring and user management.

The *Visual Wrapper Generator* is a client-server application that allows non-technical users to extract structured data from web pages without writing any code. On the client side, a user creates a set of extraction rules, called a *wrapper*, from example pages in a web browser using a point-and-click interface. The sample pages are obtained from the Content Service. When the user is satisfied with the extraction samples, the wrapper is sent to the server which applies it to similar pages in the archive. The extracted data set is stored in a relational *Extraction Database* where other users can access it through the Content Service. Figure 2 shows the GUI of the wrapper generator, which is implemented as a Google Web Toolkit application running in an ordinary browser. Section 3.2 demonstrates the extraction of a data set of book reviews using the Visual Wrapper Generator. A Flash video of the extraction process is also available [8].

Users can analyze extracted data sets in various ways using the *Visual Analysis Workbench*. This client-server application acts as a frontend to the *Map/Reduce Cluster* [1] that is used for large-scale analysis tasks on the contents of our repository. On the client side, the user specifies which data sets to process and in which way, again using a point-and-click interface (see Figure 3; a detailed description is given in Section 3.3). The visual task definition is shipped to the server side and internally translated into a declarative query language called *Marquee*[1] which is tailored to make Map/Reduce programs amenable to automatic optimization. Marquee represents the analysis task as a sequence of Map and Reduce steps that can be rearranged, e.g., in order to minimize the number of passes through the data or the size of intermediate results. Next, the Marquee query is compiled into Java code to be distributed and executed on the Map/Reduce Cluster. Before the execution begins, the specified input relations from the Extraction Database are dumped to disk and copied to the distributed file system on the cluster, too. Our current cluster has six compute nodes, each with two 2.66 GHz quad-core Xeon CPUs, 16 GB RAM and 4.5 TB local disk space, and is going to be expanded to sixty nodes soon. It runs the Hadoop [2] open-source Map/Reduce implementation. After the analysis program has terminated, all the output files it generated are moved from the cluster to disk space in the Data Repository, along with metadata about the analysis task, input/output schema, result size and runtime performance. This metadata is exposed to users that access the analysis results through the Content Service. Alternatively, the analysis results may be bulk-loaded to tables in the Extraction Database.

3. DEMO DESCRIPTION

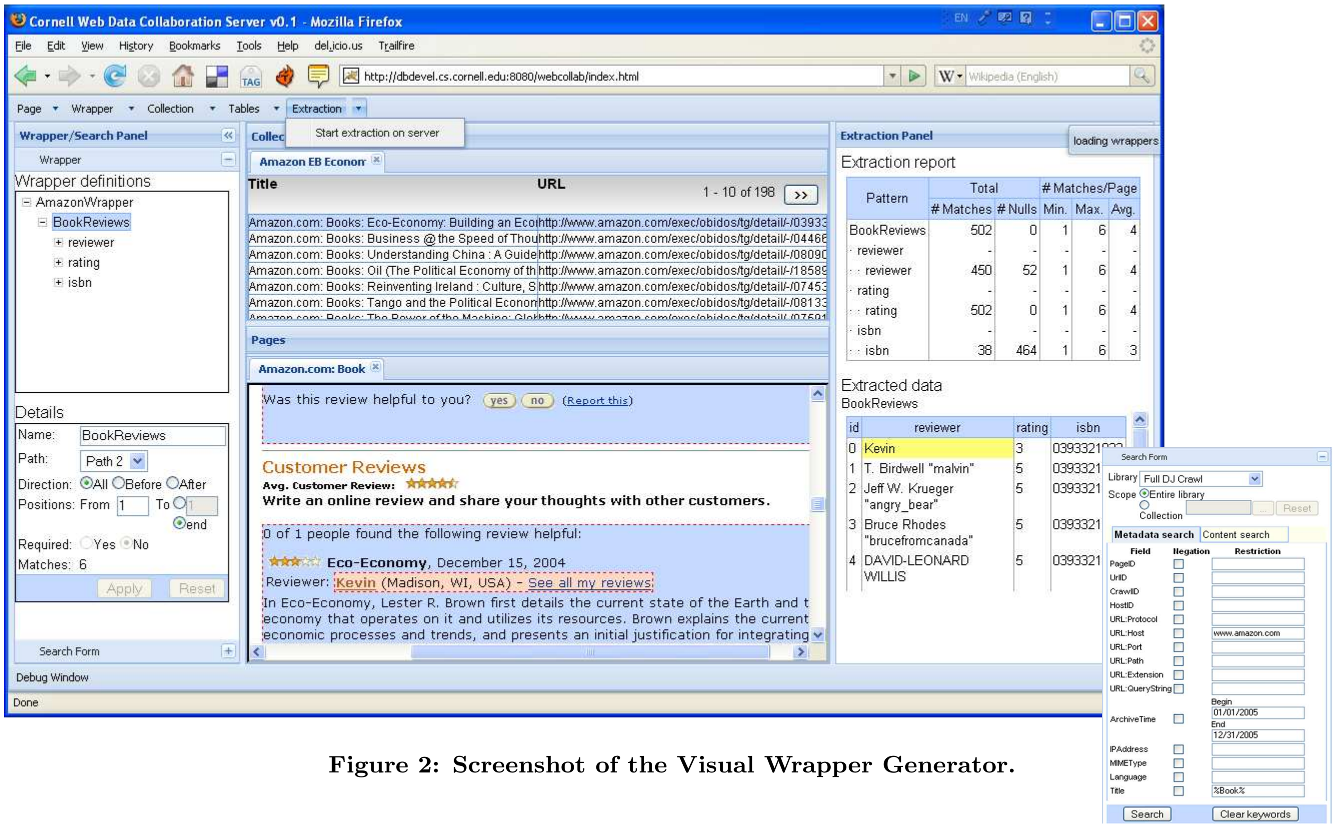
In this section, we describe a typical user session that will be presented in the demonstration, using the following example.

3.1 Example: Tie Strength in Social Networks

Research on social networks has put forward various hypotheses about the dynamics of *ties* (relations) between members of a community. For instance, according to some models, two people with strong ties to the same third person are likely to develop a strong tie between each other, too. Assume our user is a social scientist who wants to test this hypothesis on the social network of book reviewers at Amazon.com, where the strength of a tie between any two reviewers can be defined as the number of reviewed books they have in common. To compute the network structure, the user first needs to extract the ISBNs of all books in Amazon pages along with the names of reviewers who reviewed these books. Given two sets of ISBNs for a pair of reviewers, he can then obtain their tie strength as the size of the intersection of the two sets.

Figure 2 shows part of a typical Amazon web page with book reviews (bottom center). Each page describes a single

[1]*Marquee* stands for *Map/Reduce Query Language*.



Cornell Web Data Collaboration Server v0.1 - Mozilla Firefox

http://dbdevel.cs.cornell.edu:8080/webcollab/index.html

Wrapper definitions

- AmazonWrapper
  - BookReviews
    - reviewer
    - rating
    - isbn

Details

Name: BookReviews

Path: Path 2

Direction: [✓ All] Before After

Positions: From 1 To [✓ end]

Required: Yes [✓ No]

Matches: 6

Start extraction on server

Amazon EB Econom

Title	URL
Amazon.com: Books: Eco-Economy: Building an Eco	http://www.amazon.com/exec/obidos/tg/detail/-/03933
Amazon.com: Books: Business @ the Speed of Thou	http://www.amazon.com/exec/obidos/tg/detail/-/04466
Amazon.com: Books: Understanding China : A Guide	http://www.amazon.com/exec/obidos/tg/detail/-/08090
Amazon.com: Books: Oil (The Political Economy of th	http://www.amazon.com/exec/obidos/tg/detail/-/18589
Amazon.com: Books: Reinventing Ireland : Culture, S	http://www.amazon.com/exec/obidos/tg/detail/-/07453
Amazon.com: Books: Tango and the Political Econom	http://www.amazon.com/exec/obidos/tg/detail/-/08133

1 - 10 of 198

Pages

Amazon.com: Book

Was this review helpful to you? (Report this)

Customer Reviews

Avg. Customer Review: ★★★★☆

Write an online review and share your thoughts with other customers.

0 of 1 people found the following review helpful:

★★★☆☆ Eco-Economy, December 15, 2004

Reviewer: Kevin (Madison, WI, USA) - See all my reviews

In Eco-Economy, Lester R. Brown first details the current state of the Earth and t
economy that operates on it and utilizes its resources. Brown explains the current
economic processes and trends, and presents an initial justification for integrating

Extraction Panel

loading wrappers

Extraction report

Pattern	Total		# Matches/Page		
	# Matches	# Nulls	Min.	Max.	Avg.
BookReviews	502	0	1	6	4
· reviewer	-	-	-	-	-
· · reviewer	450	52	1	6	4
· rating	-	-	-	-	-
· · rating	502	0	1	6	4
· isbn	-	-	-	-	-
· · isbn	38	464	1	6	3

Extracted data

BookReviews

id	reviewer	rating	isbn
0	Kevin	3	0393321
1	T. Birdwell "malvin"	5	0393321
2	Jeff W. Krueger "angry_bear"	5	0393321
3	Bruce Rhodes "brucefromcanada"	5	0393321
4	DAVID-LEONARD WILLIS	5	0393321

Search Form

Library: Full DJ Crawl

Scope: [✓ Entire library] Collection

Metadata search | Content search

Field	Negation	Restriction
PageID	☐	
UrlID	☐	
CrawlID	☐	
HostID	☐	
URL:Protocol	☐	
URL:Host	☐	www.amazon.com
URL:Port	☐	
URL:Path	☐	
URL:Extension	☐	
URL:QueryString	☐	
ArchiveTime	☐	Begin 01/01/2005 End 12/31/2005
IPAddress	☐	
MIMEType	☐	
Language	☐	
Title	☐	%Book%

Figure 2: Screenshot of the Visual Wrapper Generator.

book, including title, ISBN, etc. Each review comes with the name of the reviewer, but no explicit mention of the book; it is understood that the review is about the book on the same page. We demonstrate the following extraction and analysis tasks:

Extraction Task: Find the subset of book pages from Amazon.com that were crawled in 2005. Extract from each page the ISBN of the book and the names of reviewers who reviewed that book.

Analysis Task: For each pair of reviewers, count the number of reviewed books they have in common and output that number as their tie strength. To simplify the network structure, include only ties of strength ≥ 3.

The analysis task can be refined in many ways, e.g., by excluding pairs of identical reviewers or by comparing the tie strengths in two crawls from different years. This can be accomplished using the techniques described below. However, for the purpose of the demonstration, we deliberately keep the example simple.

3.2 Extraction Task

This section describes the first step, the extraction of book reviews from web pages using the Visual Wrapper Generator. The GUI of the Visual Wrapper Generator is shown in Figure 2 (see [8] for an animated version). First, the user retrieves archived Amazon book pages through the metadata search form (shown in the inset on the right-hand side of Figure 2). The panel in the upper middle of the screen contains the search result as a list of hyperlinks to the repository. Clicking on an item in the list retrieves the corresponding page from the Content Service and renders it in the panel below the search result. The screenshot in Figure 2 shows an Amazon book page from 2005 with the book reviews.

While the page is being rendered, the wrapper tool dynamically adds mouse event listeners to the DOM tree that allow users to highlight and select any visible element in the page, without modifying its contents. The user first clicks on one of the reviews in the page. The tool generates an extraction rule for this element and shows the full text of that review in a table on the right-hand side. Additional columns containing the reviewer name and the ISBN of the book appear as the user selects these elements in the page. Through this immediate feedback on sample pages, users can easily check whether the extraction rules are correct. Each rule can be adjusted in the panel on the lower left, including settings for extracting attribute values or manipulating strings through regular expressions. Upon request, the tool also generalizes the extraction rule so that it captures all other reviews in the page. When the user is satisfied with the extraction result, he has the wrapper sent to the server where it is applied to all pages in the search result. The resulting `BookReviews` table is stored in the Extraction Database and registered with the Content Service. Note that since the actual extraction task (apart from sample pages) happens on the server, even large collections are processed efficiently and with a minimum of traffic between client and server.

3.3 Analysis Task

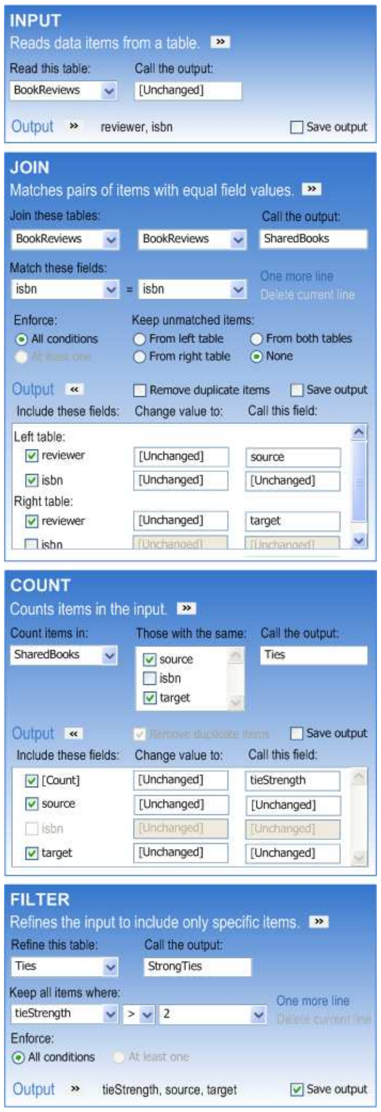
The analysis task in our example features common data processing techniques such as filtering, joining, grouping and counting. Our Visual Analysis Workbench offers a collection of predefined *analysis primitives* for these purposes that can be composed into more complex tasks. Users can also add their own tasks as new primitives to the pool, thus making them available for reuse. The following analysis primitives are currently being implemented:

Set operations: Union, intersection, difference, count, overlap (Jaccard coefficient), deduplication

Relational algebra: Selection, projection, inner and outer equijoin, grouping and aggregation

Text analysis: Word count, term frequency, document frequency, term weights (TF·IDF)

Graphs: Breadth-first search, clustering coefficient



INPUT
Reads data items from a table.

Read this table:	Call the output:
BookReviews	[Unchanged]

Output: reviewer, isbn — Save output [ ]

JOIN
Matches pairs of items with equal field values.

Join these tables:		Call the output:
BookReviews	BookReviews	SharedBooks

Match these fields: isbn = isbn

Enforce: [✓ All conditions] At least one

Keep unmatched items: From left table, From right table, From both tables, [✓ None]

Output: Remove duplicate items [ ] — Save output [ ]

Include these fields:	Change value to:	Call this field:
Left table:		
✓ reviewer	[Unchanged]	source
✓ isbn	[Unchanged]	[Unchanged]
Right table:		
✓ reviewer	[Unchanged]	target
☐ isbn	[Unchanged]	[Unchanged]

COUNT
Counts items in the input.

Count items in:	Those with the same:	Call the output:
SharedBooks	✓ source, ☐ isbn, ✓ target	Ties

Output: Remove duplicate items [✓] — Save output [ ]

Include these fields:	Change value to:	Call this field:
✓ [Count]	[Unchanged]	tieStrength
✓ source	[Unchanged]	[Unchanged]
☐ isbn	[Unchanged]	[Unchanged]
✓ target	[Unchanged]	[Unchanged]

FILTER
Refines the input to include only specific items.

Refine this table:	Call the output:
Ties	StrongTies

Keep all items where: tieStrength > 2

Enforce: [✓ All conditions] At least one

Output: tieStrength, source, target — Save output [✓]

Figure 3: User-specified analysis task in the Visual Analysis Workbench.

These primitives can be freely combined into complex analysis tasks using a form-based GUI, as shown in Figure 3. Each primitive is represented as a form with specific parameters and a description of how input tuples are processed. For example, the `INPUT` primitive reads all tuples from a user-specified table in the Extraction Database and emits them without modification. All remaining primitives consume tuples produced by other primitives. Each form has an optional section for specifying the output of the primitive. By default, such intermediate results are stored in temporary files on the analysis cluster, but they may be saved to the Extraction Database where they are available to all users.

In our example analysis (see Figure 3), the first step after accessing the `BookReviews` table is a selfjoin of that relation on the `isbn` field. The resulting relation `SharedBooks` contains the books shared by any two reviewers. The `COUNT` primitive then groups this relation by reviewer pair and emits the book count per pair as its tie strength. The final `FILTER` primitive refines this to `StrongTies` through a selection on the tie strength and saves the result in the Extraction Database.

When the user has finished editing an analysis task in the Visual Analysis Workbench, he sends it to the server which compiles it into Map/Reduce code and executes that code on the cluster. The compilation works as follows: Each visual task definition is represented internally as an expression of a *logical algebra* whose operators resemble the primitives above. The mapping between the visual and logical representation is straightforward; we skip the details in the interest of space. The logical expression is translated to a *physical algebra* expression consisting only of parameterized Map and Reduce operators that implement the various primitives. Finally, this Map/Reduce program is compiled into Java code to be linked against the Hadoop Map/Reduce runtime [2].

Composing analysis tasks in this manner is arguably simpler and more intuitive than writing either Map/Reduce programs or SQL queries by hand. Unlike procedural Map/Reduce code, our declarative analysis specification also opens up interesting opportunities for optimizing queries that combine relational operators such as selection and grouping with our extended set of primitives (see above). While on the logical level the standard query rewriting rules for relational algebra apply (e.g., selection push-down), rewriting on the physical level can alter the sequence of Map and Reduce steps in order to minimize the number of passes through the input data and the size of intermediate results. To this end, our declarative query formalism on the physical level, which we call *Marquee* (*Map/Reduce Query Language*), captures the data flow in the analysis and static properties of the Map and Reduce operators. For instance, the Marquee representation of the `FILTER` primitive in Figure 3 consists of a Map step with the property of leaving its input keys and values intact, followed by a Reduce step flagged as performing the identity mapping. Based on these properties, one possible rewriting rule merges the Map into the Reduce of the preceding `COUNT` and eliminates the Reduce of the `FILTER`, which is now obsolete. This saves the disk and network I/O needed to transfer data for another set of Map and Reduce steps. More work will be needed to develop good optimization rules and a suitable cost model for effective rewriting of general Map/Reduce programs.

4. CONCLUSION

We have introduced the Web Lab Collaboration Server, a platform and service for large-scale collaborative analysis of web data. Our goal is to turn collections of rich, but unstructured and noisy web contents into resources that are readily accessible for ad-hoc analysis in various scientific disciplines. As a first step, we have demonstrated two applications that enable users of our web archive at Cornell to write complex extraction and analysis tasks in an intuitive, integrated environment.

5. REFERENCES

[1] J. Dean and S. Ghemawat. MapReduce: Simplified Data Proc. on Large Clusters. In *Proc. OSDI*, 2004.

[2] The Hadoop Project. hadoop.apache.org.

[3] Internet Archive. www.archive.org.

[4] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. In *Proc. EuroSys*, 2007.

[5] Lixto Software GmbH. www.lixto.com.

[6] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig Latin: A Not-So-Foreign Language for Data Processing. In *Proc. SIGMOD*, 2008.

[7] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. Interpreting the Data: Parallel Analysis with Sawzall. *Scientific Programming*, 13(4):227–298, 2005.

[8] Flash video of the Visual Wrapper Generator. www.cs.cornell.edu/∼weigel/WrapperDemo.

[9] The Cornell Web Lab Project. weblab.infosci.cornell.edu.