Guided Google: A Meta Search Engine and its Implementation using the Google Distributed Web Services

Choon Hoong Ding and Rajkumar Buyya

Grid Computing and **D**istributed **S**ystems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{chd, raj}@cs.mu.oz.au

Abstract

With the ubiquity of the Internet and Web, search engines have been sprouting like mushrooms after a rainfall. However, innovative search engines and guided search capabilities have started appearing only in recent years. For instance, Google, which is one of the popular search engines, supports Web Services that allow external applications to issue Web search queries that are actually processed using a Google's commodity cluster computer made up of 15,000 PC nodes. The goals of these applications are to help ease and guide the searching efforts of novice web users towards their desired objectives. A number of implementations of such services are emerging. This paper proposes a guided meta-search engine, called *Guided Google* that serves as an advanced interface to the actual Google.com search engine.

Keywords: Web search engine, Google, Cluster, Web services, Meta-search engine, and Guided Google.

1. Introduction

In recent years, Google has grown to be one of the most popular search engines available on the Web. Like most other search engines, Google indexes Web sites, images, Usenet news groups, content-based directories, and news sources with the goal of producing search results that are most relevant to user queries. This is done using proprietary algorithms, which work based on the assumption that if a page is useful, other pages covering the similar topic are likely to provide a link to it [1] . Therefore, it can be said that Google focuses on a page's relevance and not on the number of responses.

Moreover, Google allows sophisticated searches, with required and forbidden words, and the ability to restrict results based on a particular language or encoding. However, only a small number of web users actually know how to utilize the true power of Google. Most average web users make searches based on imprecise query keywords or sentences, which returns unnecessary, or worse, inaccurate results. Based on this assumption, applications that help guide user's searching sessions have started to emerge. This is further motivated by the introduction of Google Web Services, which allows developers to query the Google server directly from their application.

Google has been providing access to its services via various interfaces such as the Google Toolbar and wireless searches. And now the company has made its index available to other developers through a Web services interface. This allows the developers to programmatically send a request to the Google server and get back a response. The main idea is to give the developers access to Google's functionality so that they can build applications that will help users make the best of Google. The Web service offers existing Google features such as searching, cached pages, and spelling correction. Currently the service is still in

beta version and is for non-commercial use only. It is provided via SOAP (Simple Object Access Protocol) over HTTP and can be manipulated in any way that the programmer pleases [2] .

This paper proposes a meta-search engine called *Guided Google* that is built using the Google Web Services. Our meta-search engine guides and allows the user to view the search results with different perspectives. This is achieved through simple manipulation and automation of Google functions that are accessible from Guided Google through the Google Web Services. It provides two functionalities: to allow "Combinatorial Keyword Searching" and "Searching by Hosts". Details of how these functions work will be discussed further in section 4 of this paper.

2. Related Work

There has been a great deal of work done in making guided searches a reality. One of the best examples is GuideBeam [3] , which is the result of research work carried out by the DSTC (Distributed Systems Technology Centre) at the University of Queensland in Brisbane, Australia. GuideBeam works based on a principle called "rational monotonicity" that emerged from artificial intelligence research in the early nineties. In the context of GuideBeam, rational monotonicity prescribes how the user's current query can be expanded in a way which is consistent with the user's preferences for information. In other words it is a guiding principle of preferential reasoning [4] . Since users prefer certain pieces of information in their quest for information, preferential reasoning fits very nicely into the picture of guided searching. Users can intuitively navigate to the desired query in a context-sensitive manner. This is known as "Query by Navigation". The goal is to elicit a more precise query from the user, which will translate into more relevant documents being returned from the associated search engine.

Another example that is more closely related to Google would be the Google API Search Tool by Softnik Technologies [5] . It is a simple but powerful Windows software tool for searching Google. It is completely free and is not meant to be a commercial product. All that the users need to do is register with Google for a license key and they will be entitled to pose 1000 queries a day. It is also an efficient research tool because it allows the users to record the search results and create reports of their research easily and automatically. The URLs, titles, etc. can be copied to the clipboard and then to a spread sheet or any other software. In summary, it enables the users to organise and keep track of their searching sessions, all at the convenience of their desktops.

The Google API Search Tool requires the users to download and install the software before they can start using it. An alternative is to have a Web-based version of the search tool. Many projects have been exploring this path and have made their tools freely available and accessible online. They include Guided Google proposed in this paper and Google API Proximity Search (GAPS) developed by Staggernation.com. The GAPS is developed using Perl and uses the Google API to search Google for two search terms that appear within a certain distance from each other on a page [6] . It does this by using a seldom-used Google feature: within a quoted phrase, * can be used as a wildcard meaning "any word". So to search for "grid**"** within 2 words of "computing", in either order, 6 queries are needed[1]:

"grid computing"
"grid * computing"
"grid * * computing"
"computing grid"
"computing * grid"
"computing * * grid"

The GAPS script simply constructs these queries, gets the first page of Google results for each query, compiles all the results, and presents them in a specified sort order. The main difference between GAPS and Guided Google is that the latter does not take into account the distance between the query terms

[1] The example given is taken from Staggernation.com

in combinatorial search. Such focused search query supported in Guided Google is likely to return more relevant response compared to the GAPS search.

It can be observed that with the introduction of Web services based access, Google has paved a way for programmers to develop applications that help users better utilize the full potential of its search engine and distributed processing infrastructure.

3. Architecture

3.1 Google Cluster

Amenable to extensive parallelization, Google's web search application lets different queries run on different processors by partitioning the overall index, also lets a single query use multiple processors [7]. To handle this workload, Google's architecture uses a cluster of more than 15,000 commodity-class PC servers with fault-tolerant software. The servers are based on Intel Architecture processors and linked using Intel® PRO/100 Ethernet adapters. The servers, which are equipped with 256MB to 1GB of RAM, run Linux operating system and have a suite of custom-built applications. This architecture makes it much more cost effective than a comparable system built out of a smaller number of high-end servers. It achieves superior performance at a fraction of the cost of a system built from fewer, but more expensive, high-end servers.

The Google cluster is further divided into smaller clusters that is made up of a few thousand machines, and is geographically distributed to protect Google against catastrophic data centric failures. Each of these smaller clusters will be assigned queries based on the user's geographic proximity to it [7, 8]. This is a form of load balancing, and it helps to minimize the round-trip time for a query; hence, giving greater response time.

The Google software architecture has been enhanced to support processing of queries raised by external third party applications, which is achieved by providing Web service interface as discussed below.

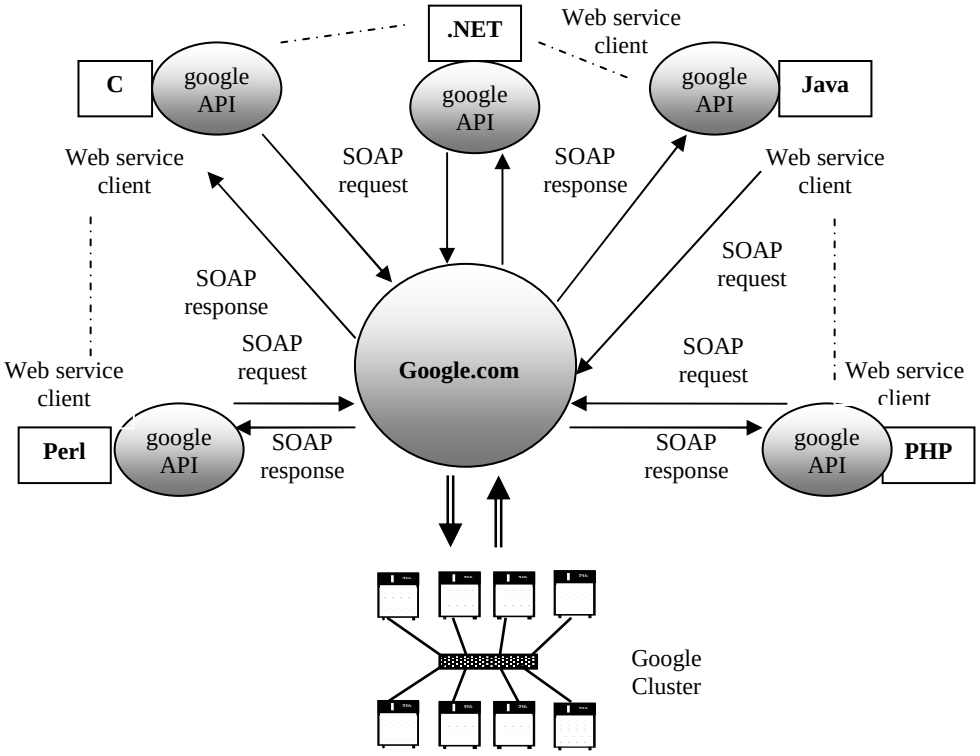

Figure 1: Google Web API

3.2 Google API

The architecture of how Google Web Services interact with user applications is shown in Figure 1. The Google server is responsible for processing users' search queries [9] . The programmers develop applications in a language of their choice (Java, C, Perl, PHP, .NET, etc.) and connect to the remote Google Web APIs service. Communication is performed via the SOAP, which is an XML (eXtensible Markup Language)-based mechanism for exchanging typed information. Once connected, the application will be able to issue search requests to Google's index of more than two billion web pages and receive results as structured data, access information in the Google cache, and check the spelling of words. Google Web APIs support the same search syntax as the Google.com site.

3.3 Guided Google Architecture

An integrated architecture of Guided Google along with its interaction between the user and the Google server is illustrated in Figure 2. The users access the Guided Google site, which is developed using JSP (JavaServer Pages) and hosted on a Tomcat server. The Tomcat server acts as both a stand-alone web server and servlet container. As seen in Figure 1, Guided Google will send and receive information from the Google.com server via SOAP. As this process is hidden, the users access it just like accessing any HTML-based web site.

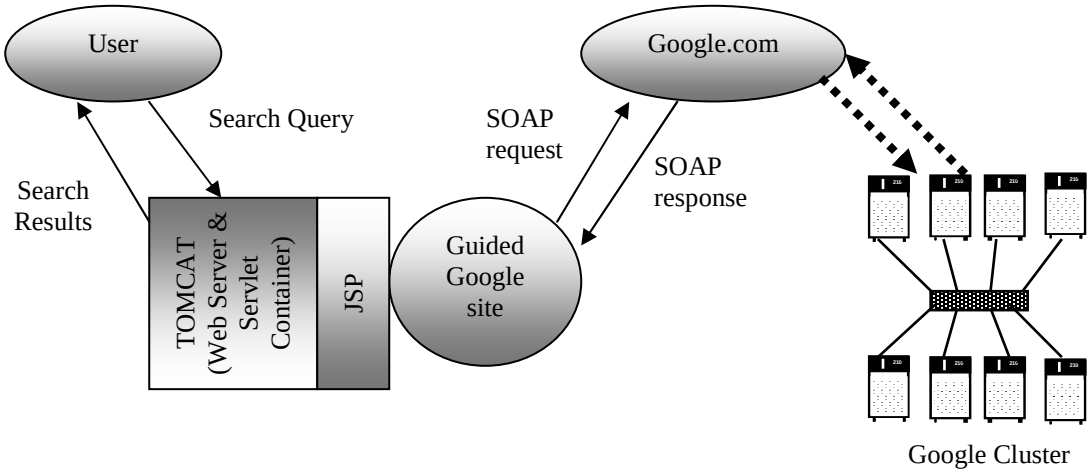

Figure 2: Guided Google Architecture.

The next section presents a detailed discussion on the design and implementation of Guided Google components.

4. Design and Implementation

Guided Google is a web based guided search tool that is developed using *Forte for Java* and coded in JSP. As mentioned earlier, this tool is developed based on the assumption that an average web user makes searches based on imprecise query keywords or sentences, which in turn leads to unnecessary or inaccurate results. In this work, we demonstrate that with a simple tweak or manipulation of existing Google functions, we can help guide users to get better search results.

The Guided Google functionalities can be grouped into two categories. The first one is for *combinatorial keyword searching*, and the second one is for *keyword searching based on the servers* hosting them. Each of these will be discussed in detail later in this section. A simple illustration of how this application is structured is discussed below.

The layered organisation of Guided Google software module is shown in Figure 3. There is a main page (index.jsp) that is used to interface with all the other .jsp files. The normal and combinatorial keyword search functions are supported by the normalSearch.jsp program file. The host based search function, on the other hand, is supported by serverBased.jsp and serverBased2.jsp program files. The first module isolates the first five unique domains from the results and the second module refines the search based on the selected host and displays the results in an expandable tree menu. Please refer to Figures 9 and 10 for a better illustration of how this works.

Other than that, several java bean components are used to support tasks that are needed to be performed multiple times. GoogleConnectionBean.java is used to establish the initial connection to the Google server. PermuterBean.java is used to calculate the permutation for the keywords when the user selects the combinatorial option. And ServerBean.java is used to store the URLs of unique domains which are obtained after the execution of the serverBased.jsp file. This allows serverBased2.jsp to access those URLs when it is executed to refine the search, based on the hosts.

To simplify the deployment, we package the entire application modules in a WAR (Web Archive) file. The hosting environment needs to be running a web server such as Tomcat 3.3 that supports JSP. The instructions for deployment of gg.war and googleapi.jar packages on the hosting server are specified in Appendix A. However, different web servers or even different versions of Tomcat may require different deployment steps.

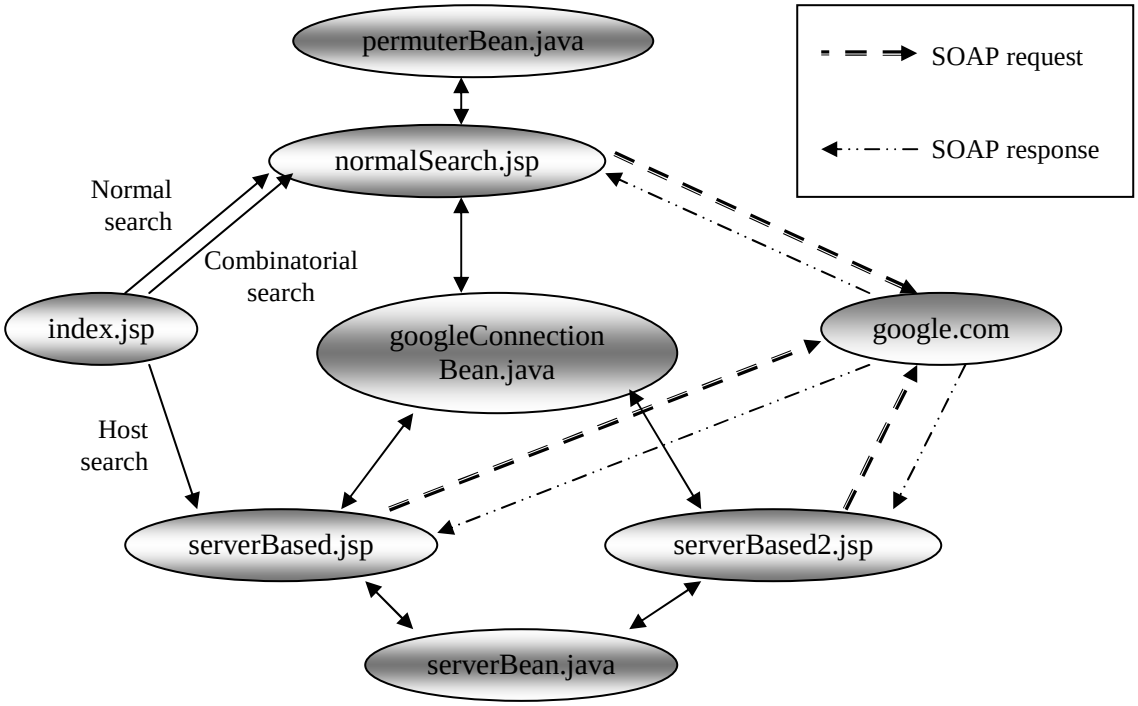

Figure 3: Layered Organisation of Guided Google Software Modules.

The remainder of this section describes the implementation and operation flow of the two functions supported in Guided Google. First of all, the entry page (as illustrated in Figure 4) consists of a query text box, a license key text box, and options as to whether the user wants to use the combinatorial or search by host functions. The users are encouraged to key in their own license key if they have registered with Google. But if they do not have one, leaving the text box 'null' would allow them to use our license key by default, which is currently limited to Google's stipulation of 1000 queries a day.

If the user submits a query keyword without selecting any of the two optional functions, Guided Google will perform a normal search (see Figure 5), as how Google.com (see Figure 6) would have done. But when one of the functions is selected, the user can get better search results as demonstrated in the next section.

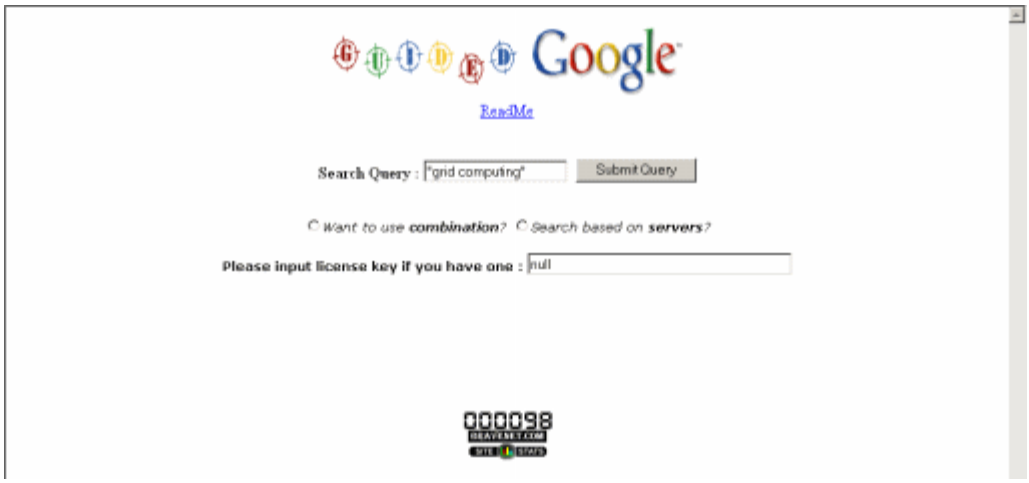

Figure 4: Guided Google entry page.

4.1 Combinatorial Search

It is widely known that the arrangement of keywords in the search query makes a difference in the search results of the Google Search Engine. Based on the assumption that novice web users are not familiar with the construction of effective keywords for their search queries, Guided Google provides a function that will automatically calculate the permutation and make different combinations of the keywords used. For example, if the users were to key in the term "grid computing", it will automatically generate two combinations, which are "grid computing" and "computing grid". The results of these two queries are very different (see Figure 7 and Figure 8). It should be noted that the quotes in search queries do make a difference. In Google search, the words in quotes mean that they have to occur in that particular order, in the search results. In accordance with this, if the search query is placed in quotes, the result of the combinations will also be reflected in quotes as demonstrated in the next section.

4.2 Search by Host

This function is slightly more complex in nature as it allows the users to search for a keyword based on the first five hosts that are hosting it. By default, Guided Google is configured to return only the first five results of a search. It is possible that there may be fewer than five hosts returned by the search. This can also happen when Guided Google automatically removes identical hosts from the list.

The 'arrow' sign shown beside each link in Figure 9 indicates that it is expandable. Clicking on the arrow triggers another query, but that will be restricted to query search within the selected host's web pages. This can be achieved by manipulating Google's query syntax (e.g., "**site:***some_host_domain*"). There are many other query syntaxes that can be very useful, depending on how the programmer manipulates them. Here, we are just showing a simple application of this in action. However, this function is not always guaranteed to give more accurate results. The main purpose of this function is to allow the user to have a different perspective of searching. It gives a lateral way of looking at the results. With the search results obtained, the users will hopefully get a better idea of what they are searching for, and hence learn to issue more accurate query keywords.

Other Google query syntax such as "inurl:", "allintext:" and "allinlinks:" can also be included. Each one of these can be manipulated in various ways to produce different outcomes.

5. Evaluation

This section focuses on evaluating the search performance of Guided Google compared to the default Google. For simplicity, we use a common search term, which is "grid computing". This term is used

throughout so that the different results can be compared. One can try for themselves how Guided Google works by following this link : http://jarrett.cs.mu.oz.au:8080/gg/index.jsp

5.1 Normal Search

In this case, "grid computing" (note that the keyword is in quotes) is submitted as the query keyword, without any options chosen, as shown in Figure 5. The results of the search are the same as a normal search done on the Google.com site (see Figure 6), except for results 3 and 5. They seem to have a different rank in the actual Google search site. Theoretically, the ranking should be the same, since we have not done any manipulation to the search. Nevertheless, the results are still the same, even though the listing sequence is slightly different in this case.

Figure 5: Normal Guided Google Search

Figure 6: Normal Google Search

5.2 Combinatorial Search

In a combinatorial search, the term "grid computing" is passed through a permutation function which gives all unique combinations of it. In this case, it will also produce "computing grid". Permutation increases in the order of factorial, so 3 keywords would mean a combination of 6 different search queries. A combinatorial search with larger number of keywords leads to exhaustive search results and it becomes hard to find relevant resources. In such situation, a normal search would have produced very accurate search results.

It should be noted that the combinatorial search function need to be used when the users are interested in more variety of results. When the users are not too sure of what they are looking for—for example, "ski holiday" or "jet boat"— they may find it worth searching in different sequences. This function automates the 'combinations portion' and searches for all the combinations. Figures 7 and 8 illustrate the results of two combinatorial searches for "grid computing". Users should use a normal search, and save them the trouble of wading through the different combination of results, if they are already sure of what they are looking for.

It can be seen in Figure 7, that the search results are the same as the normal search. Nothing has been changed there that would cause a difference in the results. Figure 8, however, shows a whole different set of returns. Although results 2 to 5 may not be in the top ten returns of the original search, they are still

relevant search results. We have made a comparison of the Figure 8 results to the results of the original search and found that result 3 is obtained in page 5, and results 2 and 4 are on page 6 of the original search. This actually gives the users a wider perspective of their search result options because users do not usually search through pages 5 and above, since most of them consider these pages irrelevant to their search. This function gives the users a chance to sample search results from different ranking levels.

Figure 7: Combinatorial Search (part 1)

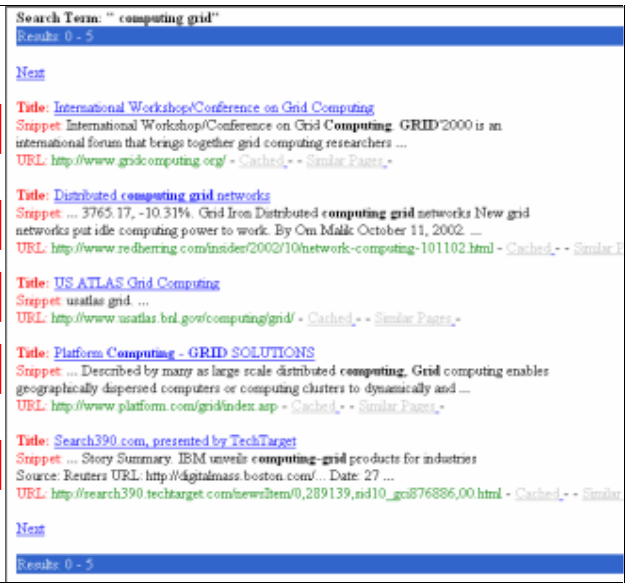

Figure 8: Combinatorial Search (part 2)

5.3 Search by Host

In searching based on hosts, the term "grid computing" is submitted to Google.com and the results are retrieved. Then the first 5 unique domains will be isolated and displayed to the user, as shown in Figure 9. Clicking on the arrows by the domains will submit another query to the Google site, (how this works was explained in the previous section) further refining the search based on the selected domain as illustrated in Figure 10.

As mentioned in the previous section, this search is not guaranteed to give better search results. But we believe that the results are just as accurate, if not better than the normal search. This is because we are picking the first 5 unique domains that are returned in the first phase of the search. Refining the search based on these domains (this is the second phase of the search) produces results that are not too far off from being accurate. In fact, searching these domains leads to more specialized search results, since we are refining the search on a domain that has already been given a high rank by Google.

The illustration in Figure 9 shows that the hosts returned are the same as the domains of the first five results of the normal search. Figure 10 shows a refined search on the "www.gridcomputingplanet.com" domain. Apart from the first result, the other four are different from the results of a normal search. These four, however, are more specific on the "grid computing" topic, since they were obtained from a single domain. The users are now able to see much more specialized results.

As illustrated, the results returned by the *'Combinatorial keyword searching'* and *'Searching by Hosts'* are very different. Even within combinatorial search, Figures 7 and 8 have highlighted the differences in results just by swapping the combination of search queries. Searching by Hosts, though it returns results that are not the highest in rank, is useful when the users want to search for a term that is related to a certain host only.

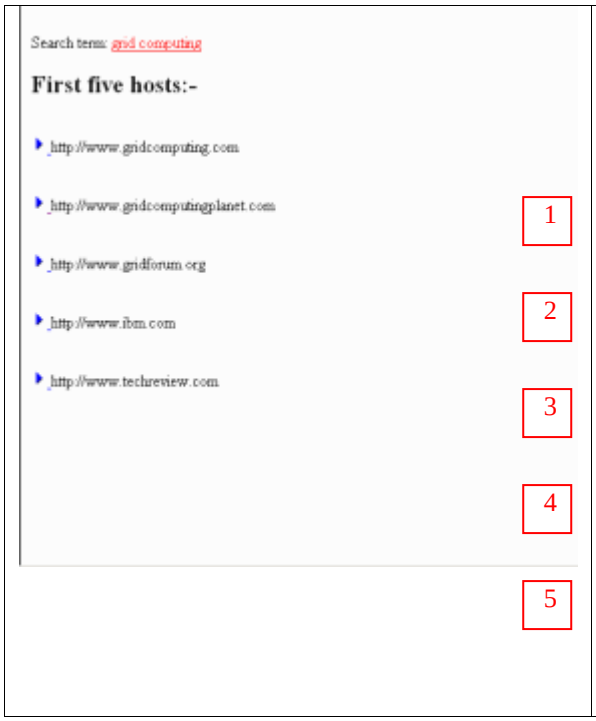

Figure 9: Search by Hosts (part 1)

Figure 10: Search by Hosts (part 2)

6. Conclusion

Modern search engines such as Google providing interfaces that allow external applications to issue Web search queries that are actually processed using their large-scale computing infrastructure such as commodity clusters. This paper proposed a guided meta-search engine, called “Guided Google”, which provides meta-search capability developed using the Google Web Services. It guides and allows the user to view the search results with different perspectives. This is achieved through simple manipulation and automation of the existing Google functions. Our meta-search engine supports search based on “combinatorial keywords” and “search by hosts”. A detailed evaluation demonstrates how one can harness the capability of Google cluster architecture through its programmable Web services by creating advanced search features at a third party user application level.

Software Availability

The Guided Google software along with source code can be downloaded from the Gridbus project web site:
http://www.gridbus.org

Acknowledgements

First and foremost, we would like to thank Google.com for developing a wonderful programming interface (Google API) that enabled us to create Guided Google. We would also like to extend our appreciation to Google Groups, for the many discussions that were held there have helped solve a few of our major problems during the implementation phase. We would like to thank Srikumar Venugopal, Chee Shin Yeo, and Anthony Sulistio for their comments on the paper.

Bibliography

[1] Sergey Brin and Lawrence Page, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, Proceedings of the 7th World Wide Web Conference (WWW7), Brisbane, Australia, April 1998. http://www-db.stanford.edu/~backrub/google.html

[2] Google Web APIs Reference - http://www.google.com/apis/reference.html

[3] Guidebeam - http://www.guidebeam.com/aboutus.html

[4] Peter Bruza and Bernd van Linder, *Preferential Models of Query by Navigation*. Chapter 4 in Information Retrieval: Uncertainty & Logics, The Kluwer International Series on Information Retrieval. Kluwer Academic Publishers, 1999. http://www.guidebeam.com/preflogic.pdf

[5] Softnik Technologies, Google API Search Tool- http://www.searchenginelab.com/common/products/gapis/docs/

[6] Google API Proximity Search (GAPS) - http://www.staggernation.com/gaps/readme.html

[7] Luiz André Barroso, Jeffrey Dean, and Urs Hölzle, *Web Search for a Planet: The Google Cluster Architecture*, Google, 2003.

[8] Mitch Wagner, *Google Bets The Farm On Linux*, June 2000, - http://www.internetwk.com/lead/lead060100.htm

[9] Ding Choon Hoong, *P2P based Content Discovery: Napster, Gnutella and Google Web API*, Technical Report, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, Dec. 2002.

[10] Google Groups (Web API) - http://groups.google.com/groups?group=google.public.web-apis

Appendix A: Guided Google Deployment Instructions (on Windows)

1. Download the appropriate jakarta-tomcat-<*version*> binary archive file.
2. Expand the archive into some directory (say C:\). This should create a new subdirectory named "jakarta-tomcat-<*version*>".
3. Set the environment variable JAVA_HOME to point to the root directory of your JDK hierarchy.
   set JAVA_HOME=c:/jdk1.3.1
   set PATH=%JAVA_HOME%\bin;%PATH%
4. Set the TOMCAT_HOME environment variable.
   set TOMCAT_HOME=c:\jakarta-tomcat-<*version*>
5. Place the gg.war file into the webapps directory (c:\jakarta-tomcat-*3.3.1/webapps)*. When Tomcat is started, it will automatically expand and deploy that file.
6. Place the googleapi.jar file into the tomcat library folder. (c:\jakarta-tomcat-3.3.1\lib\apps)
7. Start Tomcat. This is done by running the bin\startup.bat file in the bin directory of Tomcat.
8. To load the main page, type http://localhost:8080/gg/index.jsp into the browser window.
9. To stop Tomcat, just run the bin\shutdown.bat file.

Authors Biography

Choon Hoong Ding is working as a Research Assistance in the Grid Computing and Distributed Systems (GRIDS) Laboratory at the University of Melbourne, Australia. He completed Bachelor of Information Systems from Multimedia University, Malaysia and Master of Software Systems Engineering from the University of Melbourne.

Rajkumar Buyya is Director of the Grid Computing and Distributed Systems (GRIDS) Laboratory at the University of Melbourne, Australia. He has co-authored books: Microprocessor x86 Programming and Mastering C++; and edited High Performance Cluster Computing, Prentice Hall and High Performance Mass Storage and Parallel I/O, Wiley/IEEE Press. Dr. Buyya has pioneered Economic Paradigm for Service-Oriented Grid computing and demonstrated its utility through his contribution to conceptualisation, design and development of Grid technologies such as Nimrod-G, GridSim, and Gridbus.