# PERFORMANCE ANALYSIS OF METASEARCH ENGINE

**NGUYEN MINH KHOA**[1]

[1]Department of computer science and information engineering, National Taiwan University of Science and Technology, Taiwan
E-MAIL: m9815806@mail.ntust.edu.tw

**Abstract:**

**Search engines are the most useful tool for searching information on the rapidly expanding World Wide Web (WWW). In recent years, many search engines have been created to help web user finding desired information (e.g Google, Yahoo, Bing…). The fact that the search engines are providing top results that are very different from each other, if using only one search engine, internet searchers might be missing some important information. A metasearch engine is a system that provides unified access to multiple existing search engines, it allows integrating answer provided by different search engines, compare rank positions, provide advanced search features on the top of commodity search engines. In this paper, we build a simple meta-search engine to compare the results of different search engines. In addition, we also compare the time to retrieve data serially with the time to retrieve data in parallel.**

**Keywords:**

**Metasearch; world wide web; xml ; distributed information retrieval; Web;**

## 1. Introduction

The World Wide Web has become an enormous information resource in past several years. Finding desired data is one of the most popular ways the web is employed. Many search engines have been created to facilitate the retrieval of web pages. Each search engine has a text database that is defined by the set of documents that can be searched by the search engine. The fact is different search engines are providing top results are very different from each other, then by using only one search engine, internet searchers are potentially missing relevant results.

As a consequence, meta-search engine are relevant for many reasons. Metasearch engine is a system that supports unified access to multiple local search engines. It does not maintain its own index on web pages but a sophisticated metasearch engine often maintains characteristic information about each underlying local search engine in order to provide better service. When a metasearch engine receives a user query, it first passes the query (with necessary reformatting) to the appropriate local search engines, and then collects (sometimes, reorganizes) the results from its local search engines. In addition to the potential of increased search coverage of the Web, another advantage of such a metasearch engine over a general-purpose search engine is that it is easier to keep index data up to date as each local search engine covers only a small portion of the Web. In addition, running a metasearch engine requires much smaller investment in hardware (computers, storages, ...) in comparison to running a large general search engine such as Google which uses thousands of computers.

There are several serious challenges to implement an effective and efficient metasearch engine. Among the main challenges, the *database selection problem* is to identify, for a given user query, the local search engines that are likely to contain useful documents for the query. The objective of performing database selection is to improve efficiency as by sending each query to only potentially useful search engines, network traffic and the cost of searching useless databases can be reduced. In order to perform database selection well, a representative for each database needs to be stored in the metasearch engine to indicate the contents of the database.

In typical session of using metasearch engine, a user submits a query to the metasearch engine through a user friendly interface. The metasearch engine then sends the user query to a number of underlying search engines. Different component search engines may accept queries in different formats. The user query may thus need to be translated to an appropriate format for each local system. After the retrieval results from the local search engines are received, the metasearch engine merges the results into a single ranked list and presents the merged result, possibly only the top portion of the merged result, to the user.

By taking the data one by one from each search engine serially, it will be very time consuming, especially when there is so many data returned by the search engines. This paper proposed a way to take the data in parallel. By taking the data in parallel, the execution time will be increase

significantly, especially when the data is being processed are too much. But using to many parallel machines will not make it faster because of communication time from one to another.

This paper builds a simple metasearch engine to compare the retrieving result from three popular search engines (Google, Yahoo, Bing), then we can know how many results different search engines share with each other. In addition, it also compares the retrieving time using serial execution and parallel execution.

The rest of this paper is organized as follows. Section 2 describes where the datasets is taken.. In section 4 describes the system architecture. In section 4 provides the experimental results. Finally, section 5 concludes the paper with few remarks.

## 2. Datasets

The datasets are the main component to build meta-search engine. These datasets are taken from 3 established popular search engines, *Google*, *Yahoo and Bing*. Google and Yahoo allow sending the http request directly to server and get the result. While Bing requires the user registers for unique application id, then using it to create http request. Because of security reason, three search engines don't allow sending the request continuously in a short time. The http request for each search engines are shown bellow

```
[google]
#host                          =      www.google.com
host                           =      64.233.183.106
port                           =      80
desidered_pages =      2
result_per_page =      100
next_page_calc  =      2
{
        GET /search?q=%QUERYSTR&num=%RESULTS&start=%PAGE&filter=0&as_ft=e HTTP/1.0
        HOST: www.google.com
}

[yahoo]
#host                          =      search.yahoo.com
host                           =      203.69.113.25
port                           =      80
desidered_pages =      2
result_per_page =      100
next_page_calc  =      3
{
        GET /search?vf=html&va=%QUERYSTR&n=%RESULTS&b=%PAGE HTTP/1.0
        HOST: search.yahoo.com
        Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1
}

[bing]
#host                          =      api.bing.net
host                           =      163.28.5.19
port                           =      80
desidered_pages =      4
result_per_page =      50
next_page_calc  =      4
{
        GET /xml.aspx?AppId=548A7A61445B098401141A674DDD82C2B428C383&query=%QUERYSTR&
        version=2.2&Market=en-US&Sources=web&web.Count=50&web.Offset=%PAGE HTTP/1.0
        HOST: api.bing.net
```

**Figure1: HTTP request format of search engines**

## 3. System Architecture

The system is begun with the query input by users. The users submit their queries and select desired search engines among those configured in the system. This information is interpreted by the local query parser that re-writes queries in a format appropriate for each chosen engine. HTTP retrievers modules handle the network communications. As soon as search results are available, the search result collector extracts the relevant information, and format it in XML. Then the statistic and comparator extracts all the links of results and compare those links of three search engines. In addition, it calculates the execution time and some other needed indices. Lastly, the result is displayed to the user. The system architecture is illustrated in figure 2.

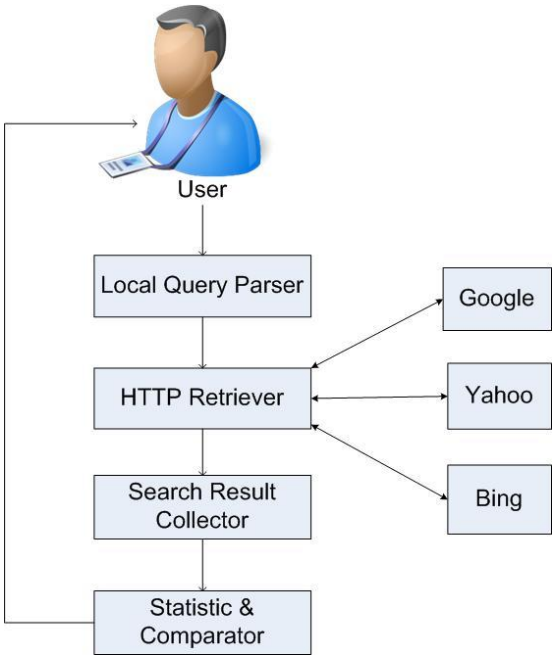


**Figure2: Meta-video search engine's architecture**

## 4. Experiments

The experiment is conducted by querying the meta-search engine with different total results and each is perform 10 times. The total results returned are 100, 200, 500. All experiments are performing on standard PCs. Because the Internet speed is the main concern of this experiment so the experiment is conducted when the Internet speed is in the highest speed.

First, the experiment for query 'computer' is

performed. The results are shown in figure3, figure4, figure5 for 100, 200, 500 results, respectively.

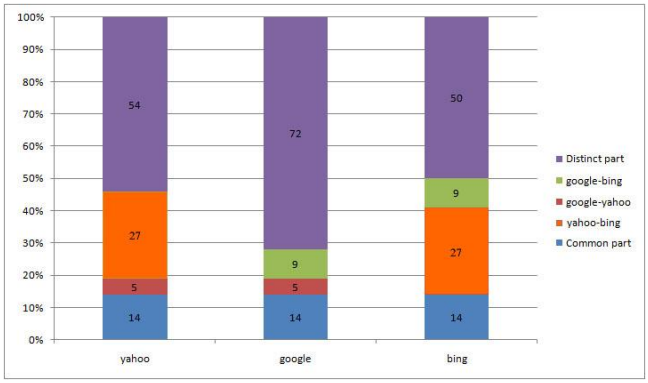


**Figure 3: Comparison of retrieving 100 results using query "computer"**

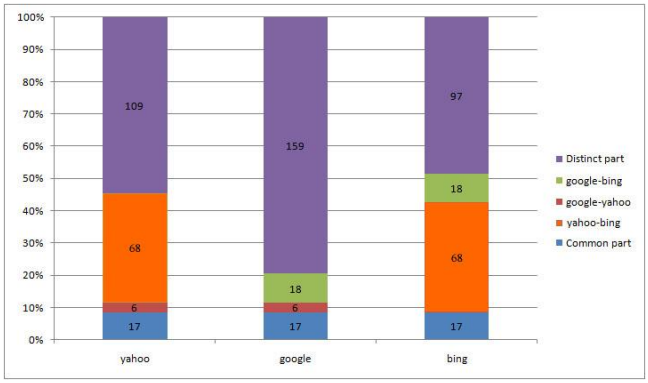


**Figure 4: Comparison of retrieving 200 results using query "computer"**

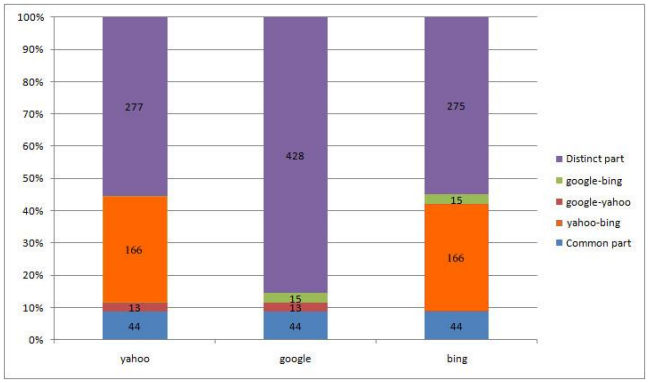


**Figure 5: Comparison of retrieving 500 results using query "computer"**

We can see that Google, Yahoo and Bing share only 14% of their top 100 results, 8.5% of their top 200 results and 8.8% of their top 500 results for query 'computer'. Yahoo and Bing share much more than each of them with Google.

Second, the experiment for top ten popular search terms is performed. The results are shown in figure6, figure7 for 100, 200 results, respectively.

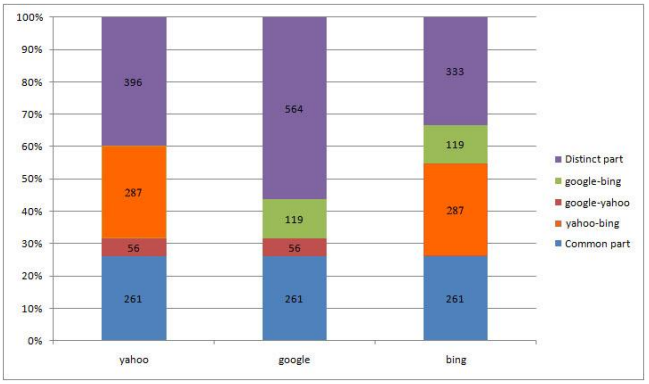


**Figure 6: Comparison of retrieving 100 results using top 10 popular queries**

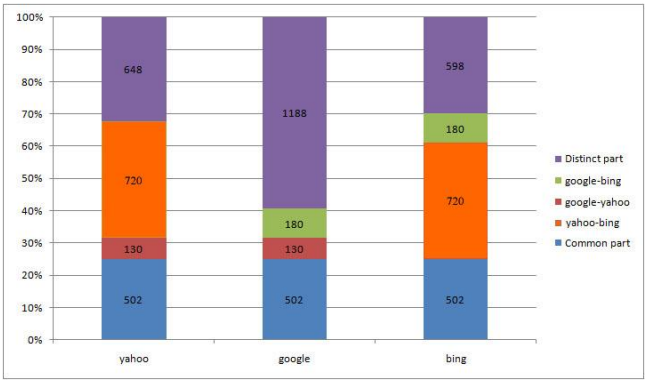


**Figure 7: Comparison of retrieving 200 results using top 10 popular queries**

We can see that Google, Yahoo and Bing share only 26% of their top 100 results, 25% of their top 200 results for top ten search terms. Again, Yahoo and Bing share much more than each of them with Google.

The third experimental is comparison between serial execution and parallel execution. We use the query "flower" and calculate the retrieving time of three search engines. The results are shown in figure8, figure9 for 100, 200 results, respectively. We can see that using parallel execution, the retrieving times are less than using serial

execution, especially when the results are 100 results, the parallel execution takes about 2.1 second and the serial execution is 3.6 second. When the results are 200 results, the parallel execution takes about 4.0 second and the serial execution is 6.4 second. The explanation for these results is very obvious, it's because the execution are done in parallel instead of serial. Then parallel execution using three HTTP connections to get the data simultaneously. Therefore, parallel execution is executed faster than serial execution. We also see that Google requires only 0.22 second to get 100 results for query 'flower', while Bing and yahoo requires 1.27 and 2.1 second, respectively.

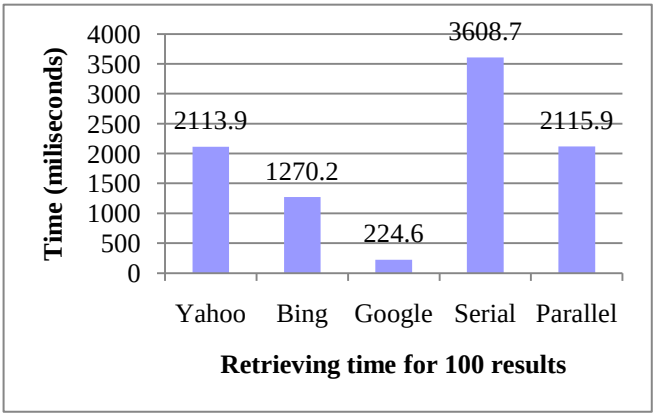


Figure 1: Comparison of retrieving time 100 results for query 'flower'

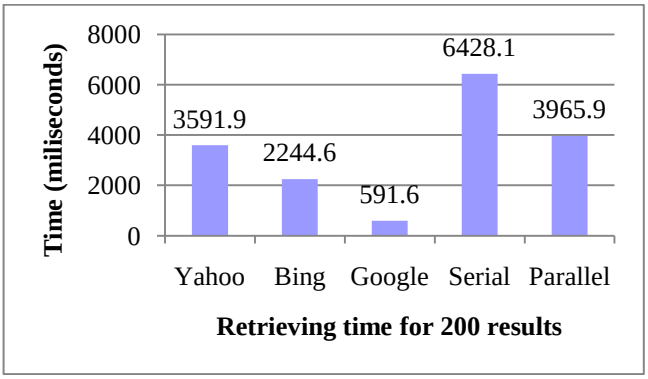


Figure 2: Comparison of retrieving time 200 results for query 'flower'

## 5. Conclusions

By building a simple meta-search engine, we perform comparison results from different search engines successfully. Google, Yahoo and Bing share only 25% for top 10 popular queries. Yahoo and Bing share more than each of them with google because Bing uses the algorithm of Yahoo. In addition, Parallel Meta-search engine is also presented. It's very practical to build the meta-search engines based on parallel execution. The reason for using this parallel is because the data are fetch in real time from the search engines providers, thus we don't need to manage the database and in other hand we can get a reasonable time to show the results to the users, yet up to date. It is must be consider not to use over parallel machines than it needs, because the communication time between the query dispatcher and the nodes must also be consider.

As far as future work is concerned, an obvious next step would be building my own merger and ranker module. By using some algorithm for raking result (eg. Borda count, Markov chain based method…), we can provide the most relevant results to user.


## Acknowledgements

This project is part of Advance Database System course in National Taiwan University of Science and Technology is taught by Professor Yi-Leh WU.


## References


[1] W. Meng, C. Yu, and K. Liu. Building efficient and effective metasearch engines. In ACM Computing Surveys, 2002.

[2] M. E. Renda and U. Straccia. Web metasearch: Rank vs. score basedrank aggregation methods. In SAC, 2003.

[3] Zonghuan Wu, Weiyi Meng, Clement Yu, and Zhuogang Li. Towards a Highly-Scalable and Effective Metasearch Engine. *Proc. of Tenth World Wide Web Conference (WWW10)*, Hong Kong, May 2001, pp.386-395.

[4] Bing Developer Center. [Online] Available at: http://www.bing.com/developers

[5] Yahoo! Developer Network. [Online] Available at: http://developer.yahoo.com/

[6] Google Web Search API. [Online] Available at: http://code.google.com/more/