Google Search Engine Notes

These notes are for the 1998 prototype of the Google large-scale search engine. Google's name is derived from the common spelling of googol--the large number 10^{100}.

History

In 1998, Yahoo was the leading search engine. Yahoo used a manual approach to have high quality indices. As a result, popular topics were subjective, expensive to build, slow to update, and did not cover obscure topics.

Automated search engines returned low quality matches. Advertisers were able to manipulate their Web pages to mislead these engines. Users were accustomed to having to look through at least ten results to find a single relevant result.

Design Goals

1. **Web Scale.** At the rate the Web was growing, fast crawling is required to be able to process hundreds of gigabytes efficiently. Search queries need to be handled quickly at a throughput of thousands of queries per second.
2. **Search Quality**. Users should only receive the most relevant results.
3. **Academic Research**. Current search engines cater to commercial entities and advertisers. Researchers need a means to process large portions of the Web to produce meaningful, academic results.

PageRank

PageRank utilizes link structure and anchor text (the visible, clickable text in a hyperlink) for relevance and quality filtering. PageRank is a model of user behavior. The quality and importance of a page can be measured primarily by counting the number of links from other pages. Pages that have a high PageRank can be considered to have higher quality links. PageRank has many tunable parameters. It is designed so that a single factor cannot have too much influence on the overall rank.

Anchor Text

Research has found that anchor text provides the most accurate description of Web pages, more so than the Web pages' description of itself. Using anchor text allows for higher quality categorizations of Web pages.

Anchor text allows for categorization of non-text content as well e.g., images, videos, programs, and databases. Non-text content cannot be indexed by a standard text-based search engine. Anchor text from other pages opens up that possibility.

Anchor text expands the search coverage of downloaded documents. In Google's crawl of 24 million pages, more than 259 million anchors were indexed--about 10x more coverage.

Google Architecture

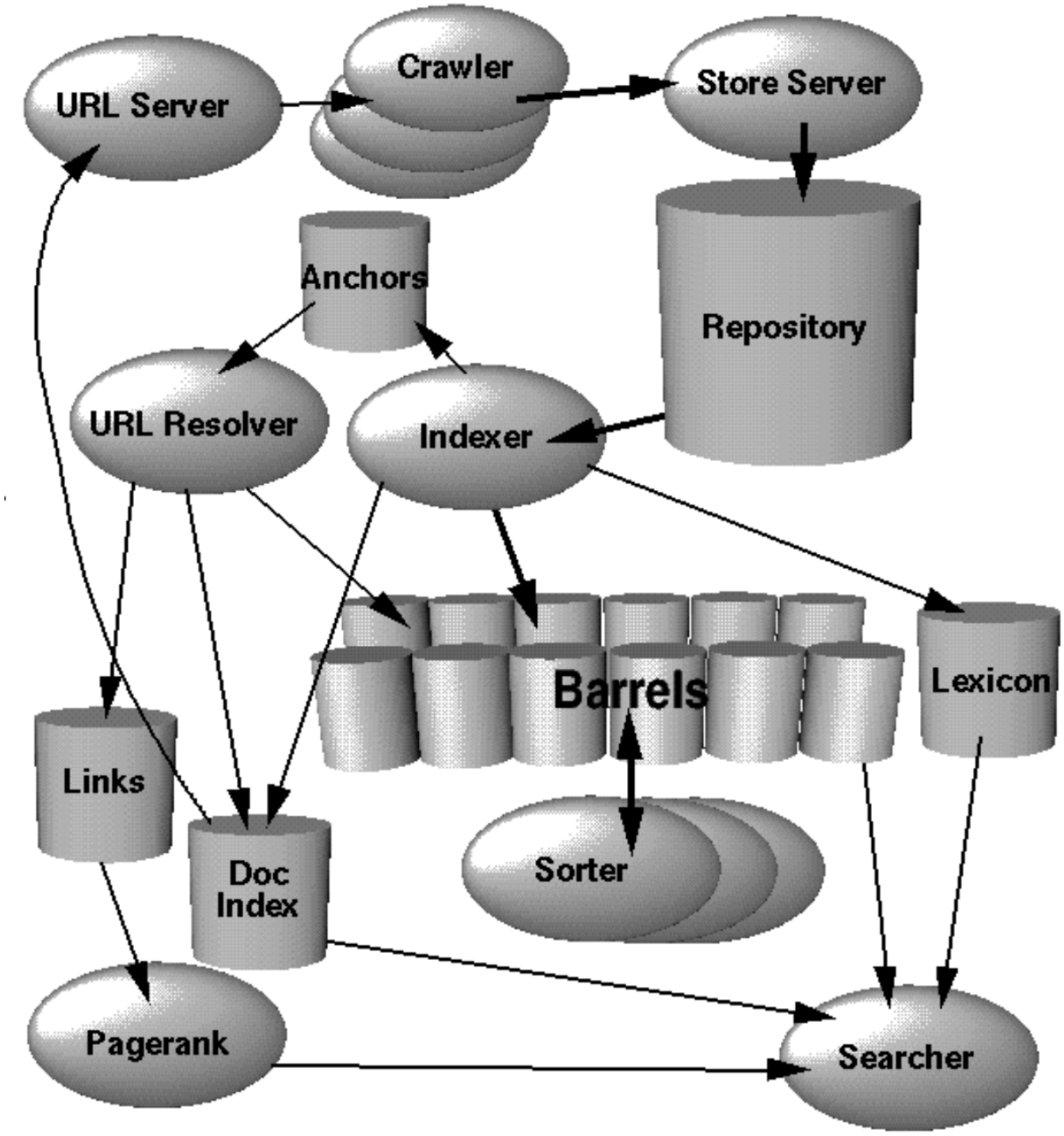

Figure 1. High Level Google Architecture

Most of Google is implemented using C/C++ in a Linux environment.

1. **URL Server**. The *URL Server* sends lists of URLs to be fetched to the *Crawlers*. Other systems refer to the *URL Server* as the *URL Frontier*.
2. **Crawlers**. Several distributed *Crawlers* are responsible for downloading Web pages.

3. **Store Server**. The *Store Server* compresses and stores the fetched Web pages into a repository. Each Web page is assigned a `docID`.
4. **Indexer.** The *Indexer* gets documents from the repository. Next, it uncompresses and parses them. For each document, hits (a set of word occurrences) are generated. The word, its position in the document, font size, and capitalization are recorded. The *Indexer* distributes these hits into *Barrels*--this is the forward index. The forward index is sorted by `docID`. Also, the *Indexer* gathers all the anchors from the documents and places them into an *Anchors* file.
5. **URL Resolver.** The *URL Resolver* reads the *Anchors* file and converts relative URLs into absolute URLs. Each URL is assigned a `docID`. It adds anchor text to the forward index with the associated `docID`. Also, the *URL Resolver* places the links paired with its `docID` into the *Links* database.
6. **PageRank.** *PageRank* uses the *Links* database to compute PageRanks for the documents.
7. **Sorter.** The *Sorter* takes the forward index from the *Barrels*. Each word is assigned a `wordID`. The *Sorter* re-sorts the forward index by `wordID` in place--this creates the inverted index.
8. **DumpLexicon.** The *DumpLexicon* takes the results from the *Sorter* and *Indexer* to generate a new lexicon to be used by the *Searcher*.
9. **Searcher**. The *Searcher* uses the new lexicon with the inverted index and PageRanks to provide results to queries.