The Anatomy of a Large-Scale Hypertextual Web Search Engine -Google

Author: Sergey Brin and Lawrence Page

Presented by   Xiangjun Qin
               Yanhui Ma
Overview

• Introduction – Web Search Engine
• Design Goals of Google
• System Features
• System Anatomy of Google
  – Architecture, Data Structure, Web Crawling, Indexing, Searching
• Performance
• Conclusion
Introduction – Web Search Engine

- WWW – 1994 – Index 110k web – 1500q/d
- Web Crawl – 1997 – Index 2-100M web – 20M q/d
- Google – 2000 – Index a billion web –

- Search Engine Design
  Make it easy to find everything --> Best quality
Google – Design Goal

- Improve the search quality: to build a high precision tool to include only the very best document
- Push more development and understanding into the academic realm
- Build systems that many people can use
- Build an architecture that can support novel research activities on large-scale web data
System Features

• PageRank: an objective measure of citation importance of a web document
  – $\text{PR}(A) = (1-d) + d \left( \frac{\text{PR}(T_1)}{\text{C}(T_1)} + \ldots + \frac{\text{PR}(T_n)}{\text{C}(T_n)} \right)$
  – A page has a high PA is there are many pages pointing to it, or some of those pages have high PA

• Anchor Text:
  – Associate text of a link with the page that the link is on
  – Associate text with page the link points to
Related Work

• Information Retrieval
  – Standard information retrieval work needs to be extended to deal effectively with web

• Difference between Web and Well Controlled Collections
  – Docs on the web have extreme variation internal to the docs, and also in the external meta information
  – There is no control over what people can put on web
System Anatomy – Google Architecture
System Anatomy – Google Architecture (cont’)

- **Web Crawling**: Download web pages
- **URLserver**: send URLs to crawlers
- **StoreServer**: compress pages
- **Repository**: store pages
- **Indexer**: read and parse docs, convert docs to hits, distribute hits into barrels, parse out all links
- **URLSolver**: convert URLs and turn into docIDs, and create forward index
System Anatomy – Google Architecture (cont’)

- **Sorter**: take barrels, resort them by wordID, generate inverted index, produce list of wordID
- **DumpLexicon**: take the list and generate new lexicon
- **Searcher**: use the lexicon, inverted index and PageRanks to answer queries
System Anatomy – Data Structure

- **Repository**
  - Contains full HTML of every web page
  - Use zlib to compress

- **Document Index**
  - Fixed width ISAM index, ordered by docID

- **Lexicon**
  - 14 million words and can be fit into memory
  - Implement as a list of words and a hash table of pointers
System Anatomy – Data Structure

- **Hits Lists:**
  - A list of occurrences of a particular word in a particular document
  - Fancy hits and plain hits
- **Forward and Inverted Indexes:**

  **Hit:** 2 bytes

  | plain: cap:1  | imp:3  | position: 12 |
  | fancy: cap:1 | imp = 7 | type: 4 | position: 8 |
  | anchor: cap:1 | imp = 7 | type: 4 | hash: 4 | pos: 4 |

  **Forward Barrels:** total 43 GB

<table>
<thead>
<tr>
<th>docid</th>
<th>wordid: 24</th>
<th>nhits: 8</th>
<th>hit</th>
<th>hit</th>
<th>hit</th>
<th>hit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>wordid: 24</td>
<td>nhits: 8</td>
<td>hit</td>
<td>hit</td>
<td>hit</td>
<td>hit</td>
</tr>
<tr>
<td></td>
<td>null wordid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>docid</th>
<th>wordid: 24</th>
<th>nhits: 8</th>
<th>hit</th>
<th>hit</th>
<th>hit</th>
<th>hit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>wordid: 24</td>
<td>nhits: 8</td>
<td>hit</td>
<td>hit</td>
<td>hit</td>
<td>hit</td>
</tr>
<tr>
<td></td>
<td>wordid: 24</td>
<td>nhits: 8</td>
<td>hit</td>
<td>hit</td>
<td>hit</td>
<td>hit</td>
</tr>
<tr>
<td></td>
<td>null wordid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  ... 

  **Lexicon:** 293 MB

  **Inverted Barrels:** 41 GB

<table>
<thead>
<tr>
<th>wordid</th>
<th>ndocs</th>
</tr>
</thead>
<tbody>
<tr>
<td>wordid</td>
<td>ndocs</td>
</tr>
<tr>
<td>wordid</td>
<td>ndocs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>docid</th>
<th>nhits: 5</th>
<th>hit</th>
<th>hit</th>
<th>hit</th>
<th>hit</th>
</tr>
</thead>
<tbody>
<tr>
<td>docid</td>
<td>nhits: 5</td>
<td>hit</td>
<td>hit</td>
<td>hit</td>
<td>hit</td>
</tr>
<tr>
<td>docid</td>
<td>nhits: 5</td>
<td>hit</td>
<td>hit</td>
<td>hit</td>
<td>hit</td>
</tr>
<tr>
<td>docid</td>
<td>nhits: 5</td>
<td>hit</td>
<td>hit</td>
<td>hit</td>
<td>hit</td>
</tr>
</tbody>
</table>

  ...
Crawling and Indexing Web

• Crawling Web:
  – Google has a fast distributed crawling system
  – 100 web pages/s at peak speed

• Indexing Web:
  – Parsing with flex to generate lexical analyzer
  – Indexing documents into barrels
  – Sorting each forward barrels by wordID to produce an inverted barrel
Searching – Google Query Evaluation

1. Parse the query.
2. Convert words into wordIDs.
3. Seek to the start of the doclist in the short barrel for every word.
4. Scan through the doclists until there is a document that matches all the search terms.
5. Compute the rank of that document for the query.
6. If we are in the short barrels and at the end of any doclist, seek to the start of the doclist in the full barrel for every word and go to step 4.
7. If we are not at the end of any doclist go to step 4.
   Sort the documents that have matched by rank and return the top k.
Results and Performance

Query: bill clinton

http://www.whitehouse.gov/
100.00%  (no date) (0K)
http://www.whitehouse.gov/Office of the President
99.67%  (Dec 23 1996) (2K)
Welcome To The White House
99.98%  (Nov 09 1997) (5K)
http://www.whitehouse.gov/WH/Welcome.html
Send Electronic Mail to the President
99.86%  (Jul 14 1997) (5K)
http://www.whitehouse.gov/WH/Mail/html/Mail_President.html

mailto:president@whitehouse.gov
99.98%
mailto:President@whitehouse.gov
99.27%
The "Unofficial" Bill Clinton
94.06%  (Nov 11 1997) (14K)
http://zpub.com/un/un-bc.html
Bill Clinton Meets The Shrink
86.27%  (Jun 29 1997) (63K)
President Bill Clinton - The Dark Side
97.27%  (Nov 10 1997) (15K)
http://www.realchange.org/clinton.htm
$3 Bill Clinton
94.73%  (no date) (4K)
http://www.gateway.net/~tjohnson/clinton1.html
## Results and Performance (Statistics)

<table>
<thead>
<tr>
<th>Storage Statistics</th>
<th>Web Page Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Size of Fetched pages</td>
<td>Number of Web Pages Fetched</td>
</tr>
<tr>
<td>147.8GB</td>
<td>24 million</td>
</tr>
<tr>
<td>Compressed Repository</td>
<td>Number of URLs Seen</td>
</tr>
<tr>
<td>53.5 GB</td>
<td>76.5 million</td>
</tr>
<tr>
<td>Short Inverted Index</td>
<td>Number of Email Addresses</td>
</tr>
<tr>
<td>4.1 GB</td>
<td>1.7 million</td>
</tr>
<tr>
<td>Full Inverted Index</td>
<td>Number of 404’s</td>
</tr>
<tr>
<td>37.2 GB</td>
<td>1.6 million</td>
</tr>
<tr>
<td>Lexicon</td>
<td></td>
</tr>
<tr>
<td>293 MB</td>
<td></td>
</tr>
<tr>
<td>Temporary Anchor Data</td>
<td></td>
</tr>
<tr>
<td>6.6 GB</td>
<td></td>
</tr>
<tr>
<td>(not in total)</td>
<td></td>
</tr>
<tr>
<td>Document Index Incl. Variable Width Data</td>
<td></td>
</tr>
<tr>
<td>9.7 GB</td>
<td></td>
</tr>
<tr>
<td>Links Database</td>
<td></td>
</tr>
<tr>
<td>3.9 GB</td>
<td></td>
</tr>
<tr>
<td><strong>Total Without Repository</strong></td>
<td></td>
</tr>
<tr>
<td><strong>55.2 GB</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Total With Repository</strong></td>
<td></td>
</tr>
<tr>
<td><strong>108.7 GB</strong></td>
<td></td>
</tr>
</tbody>
</table>
Conclusions and Future Work

• Google is scalable search engine to provide high quality search results
• Google employs techniques with page rank, anchor text to improve search quality
• Google is a complete architecture for gathering web pages
• Future work:
  – Boolean operators, negation and stemming
  – Relevance feedback and clustering
  – User context and result summaries