The Anatomy of a Large-Scale Hypertextual Web Search Engine

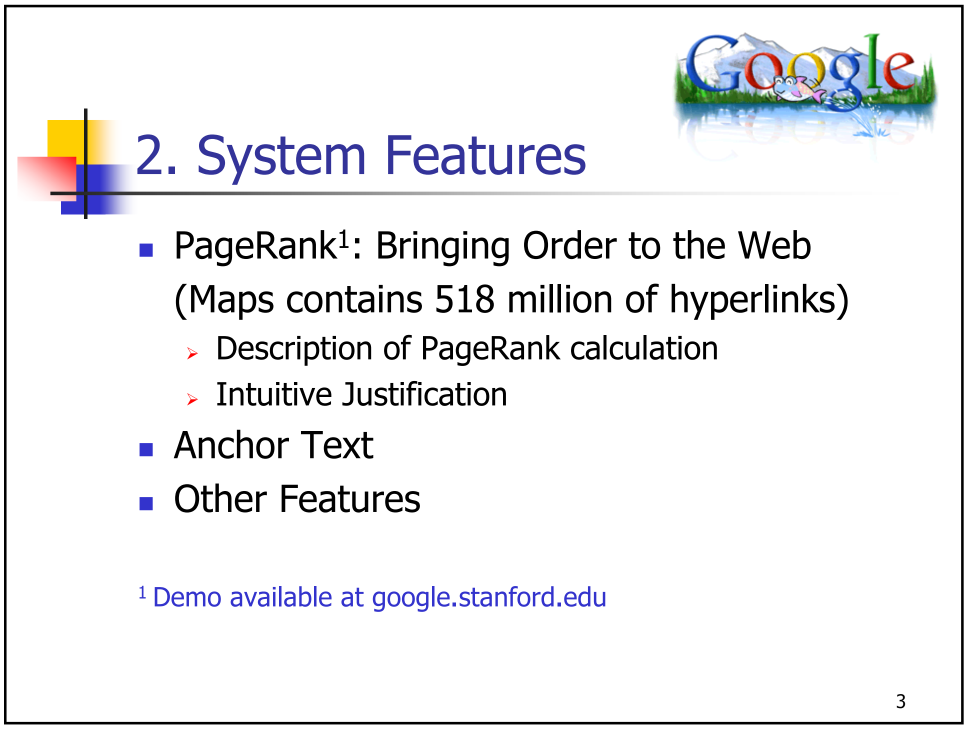
Article by: Larry Page and Sergey Brin
Computer Networks 30(1-7):107-117, 1998

1. Introduction

- The authors: Lawrence Page, Sergey Brin
  started small at Stanford Univ. during their graduate studies

- Google index grew to over 1 billion pages in June 2000
- At the time, Google served an average of 18 million queries/day
- Google index grew to over 8 billion pages in 2003

- Traditional keyword searches not accurate enough
- Google makes use of html document structure

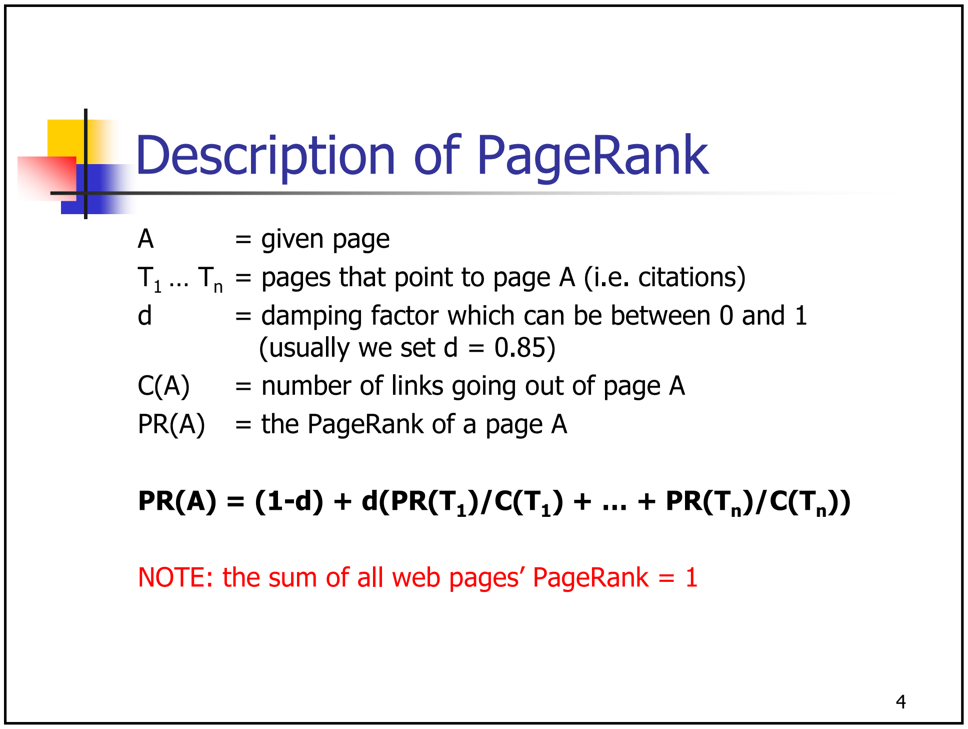


2. System Features

- PageRank[1]: Bringing Order to the Web (Maps contains 518 million of hyperlinks)
  - Description of PageRank calculation
  - Intuitive Justification
- Anchor Text
- Other Features

[1] Demo available at google.stanford.edu

3



Description of PageRank

A	= given page
$T_1 \ldots T_n$	= pages that point to page A (i.e. citations)
d	= damping factor which can be between 0 and 1 (usually we set d = 0.85)
C(A)	= number of links going out of page A
PR(A)	= the PageRank of a page A

$$\mathbf{PR(A) = (1\text{-}d) + d(PR(T_1)/C(T_1) + \ldots + PR(T_n)/C(T_n))}$$

NOTE: the sum of all web pages' PageRank = 1

4

Intuitive Justification

- Assume there is a "random surfer"
- Probability that random surfer visits a page is its PR
- 1-d is the probability at each page that the "random surfer" will get bored
- Variations of the formula
- A page has high rank:
  - If there are many pages that point to it
  - If there are some pages that point to it and have a high PageRank

5

Anchor Text

- Associate the text of a link with the page that the link is on and the page the link points to
- Advantages:
  - Anchors often provide more accurate description
  - Anchors may exist for documents which cannot be indexed (i.e. image, programs, and databases)
- Propagating anchor text helps provide better quality results
- 24 million pages → over 259 million anchors

6

Other Features

- Location information for all hits
- Keep track of some visual presentation details (i.e. font size of words)
- Full raw HTML of pages is available in a repository

7

3. Related Work

- Information Retrieval
  - In the past, focused largely on scientific stories, articles, etc.
  - Ex. Assignment 1 (Vector Space Model)
- Differences between web and controlled collections
  - The web contains varying content (html, doc, PDF, etc.)
  - No control of web content

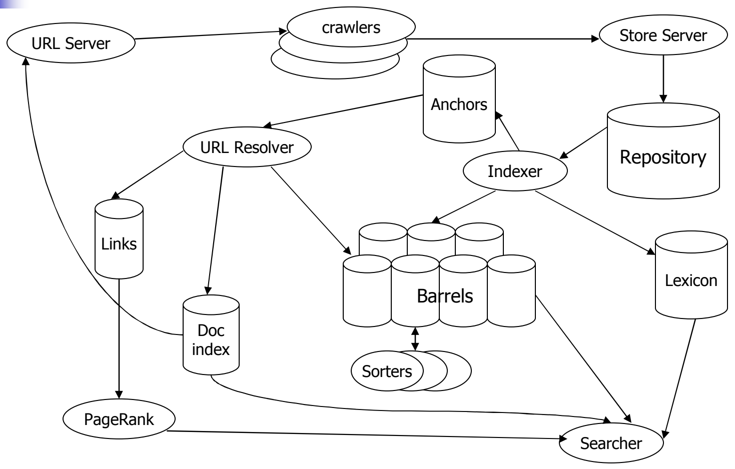
8

4. System Anatomy

- Provide a high level discussion of the architecture
- Some in-depth description of important data structure
- The major components:
  - crawling
  - indexing
  - Searching
- Implemented in C/C++ for Linux/Solaris

9

Google Architecture overview



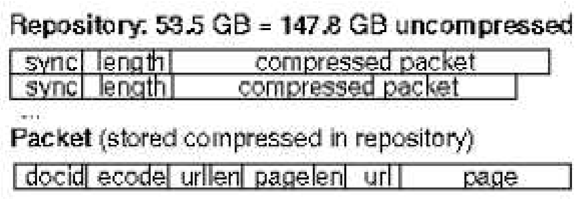
10

Major Data Structure

- BigFiles
  - Virtual files are addressable by 64 bit integers
  - Support rudimentary compression options
- Repository
  - Contains full HTML of every web page
  - Compression rate of zlib is 3 to 1 compare to bzib is 4 to 1
  - docID, length, URL
- Document Index
  - Keep info of each document (docID, fixed width ISAM index)
  - Current doc status, a pointer into the repository, a doc checksum, and various statistics

11

Repository data structure



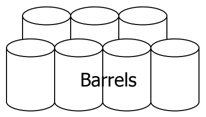
12

Major Data Structure (cont’d)

- Lexicon
  - Repository of words
  - Implemented with a hash table of pointers (word ids) to barrels (that are sorted lists)
  - 14 million words, plus an extra file for rare words
- Hit Lists
  - Stores occurrences of a particular word in a particular document
  - Types of hits: Plain, Fancy and anchor (2 bytes per hit)

13

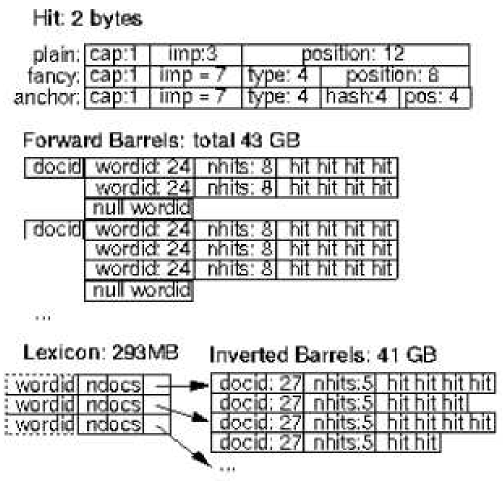
Major Data Structure (cont’d)



- Forward Index
  - Barrel holds a range of word ids
  - Doc id and a list of word ids
- Inverted Index
  - Similar to Assignment 1 inverted index
  - Can be sorted by **doc id** or by **ranking** of word occurrence

14

Forward and Inverted Index and the Lexicon



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
docid wordid: 24 nhits: 8 hit hit hit hit
wordid: 24 nhits: 8 hit hit hit hit
null wordid
docid wordid: 24 nhits: 8 hit hit hit hit
wordid: 24 nhits: 8 hit hit hit hit
wordid: 24 nhits: 8 hit hit hit hit
null wordid
Lexicon: 293MB
wordid ndocs
wordid ndocs
wordid ndocs
Inverted Barrels: 41 GB
docid: 27 nhits:5 hit hit hit hit
docid: 27 nhits:5 hit hit hit
docid: 27 nhits:5 hit hit hit hit
docid: 27 nhits:5 hit hit

15

Crawling the Web

- Crawlers need to be reliable, fast and robust
- Some authors don't want their pages to be crawled
- Google (1998) had about 3-4 crawlers running
- At peek speeds, the 4 crawlers processed 100 web pages/sec.
- Crawlers have different states: DNS lookup, connecting to host, send request, receiving response
- Crawlers and URL server were implemented in Python

16

Indexing the Web

- Parsing
  - Parsers need to handle errors very well (typos, formatting, etc.)
- Indexing
  - After parsing, placed into forward barrels
  - Words converted into word id and occurrences into hit lists
- Sorting
  - Forward barrels are sorted by word id to produce an inverted index

17

Searching

- Goal of searching: quality search results efficiently
- Ranking system:
  - Every hit list includes position, font and capitalization info
  - Consider each hit to be one of several different type and each of which has its own type-weight
  - Many parameters: type-weight, type-prox-weight (for phrasal queries)
- User feedback mechanism

18

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.

19

5. Results and Performance

- Storage Requirements
  - Google (1998) compressed its repository to about 53GB (24 million pages)
  - Additional storage used by Google for indexes, temporary storage, lexicon, etc. required about 55GB.

20

System Performance

- Crawling & Indexing efficiently
  - Crawling took a week or more
  - To improve efficiency, the Indexer and Crawlers need to be capable of simultaneous execution
  - Sorting needs to be done in parallel on several machines

21

Search Performance

- Disk I/O – Requires most time (seek time, transfer time, network delay, etc.)
- Caching – Reduces the number of disk access
- Performance times can improve from 2.13 sec. to 0.06 sec.

22

6. Conclusion

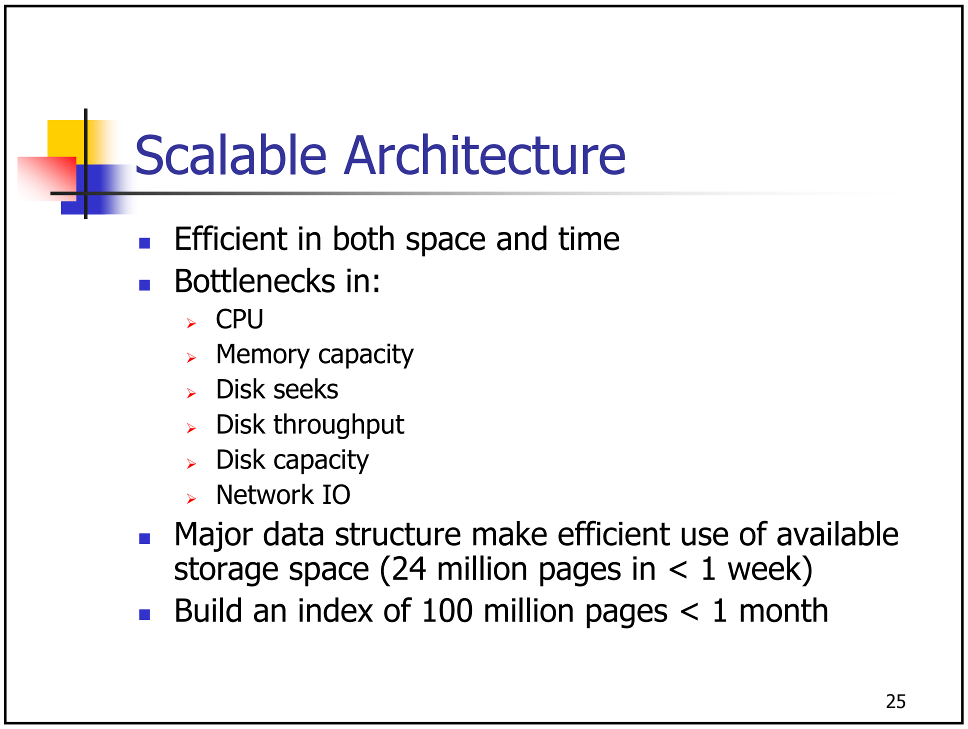
- Designed to be a scalable search engine
- Provide high quality search
- Complete architecture for gathering web pages, indexing them, and performing search queries over them

23

High Quality Search

- Heavy use of hypertextual info
  - Link structure
  - Link (anchor) text
- Proximity and font info → increase relevance
- PageRank → quality
- Link text → relevant

24



Scalable Architecture

- Efficient in both space and time
- Bottlenecks in:
  - CPU
  - Memory capacity
  - Disk seeks
  - Disk throughput
  - Disk capacity
  - Network IO
- Major data structure make efficient use of available storage space (24 million pages in < 1 week)
- Build an index of 100 million pages < 1 month

25