



- Authors: Sergey Brin, Lawrence Page
- Google, word play on googol or 10^{100}
- Centralized system, entire HTML text saved
- Focused on high precision, even at expense of high recall
- Relies heavily on document link structure:
 - Backlinks
 - PageRank

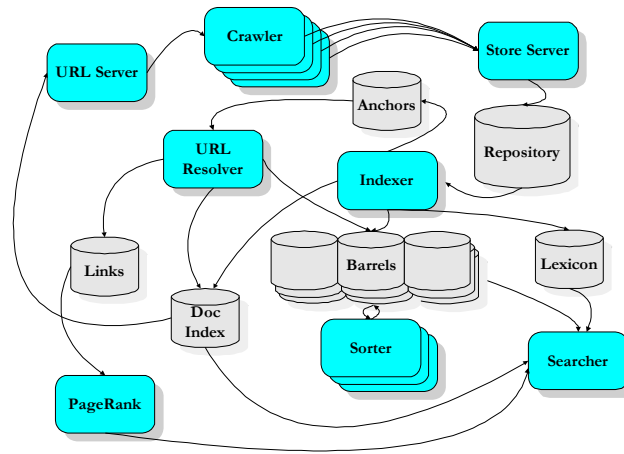


Size

- “Google operates what is probably the world's largest Linux cluster that puts many supercomputing centers to shame. For example, the current cluster contains 80 TB of disk storage and has an aggregate I/O bandwidth of about 50 GB/sec (that's bytes, not bits). “

Source: Google.com Employment Page

Architecture

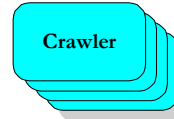


URL Server

URL Server

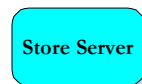
- Takes an initial starting URL as input
- Reads URLs from Document Index
- Sends URL's to Crawlers for fetching
- URL ordering is important (depth-, breadth-first)

Crawlers



- Fetches pages from the web
- Stores pages on Store Server
- Google utilizes 4 crawlers with approximately 300 connections each. Peak capacity about 100 pages per second (600 kiloBytes)
- Robots Exclusion Protocol - mechanism for web authors to mark portions of a site that should not be crawled. Not an official standard, a consensus among robot denizens

Store Server



- Compresses pages into zlib (RFC1950) format
- Approximately 3:1 compression
- Receive raw pages from the crawlers
- Stores pages into the repository
- Contains a unique docID for each web page

Indexer

Indexer

- Reads repository and uncompresses docs
- Parses each doc, converts each word into “hits”
 - Word
 - Position in document
 - Approximation of font size (relative to rest of doc)
 - Capitalization
- Fancy hits: hits occurring in an URL, title, anchor text or meta tag. All other hits are plain hits.

Indexer (cont.)

Indexer

- Distributes hits into barrels creating a partially sorted forward index:

Forward Index: 43GB

docID	wordID	numhits	hit	hit	hit	hit
	wordID	numhits	hit	hit	hit	hit
	null wordID					
docID	wordID	numhits	hit	hit	hit	hit
	wordID	numhits	hit	hit	hit	hit
	wordID	numhits	hit	hit	hit	hit
	null wordID					

←24 bits → ←8 bits →

- Parses out links and stores information (fromURL, toURL, text of link) into anchors file
- Generates lexicon file

URL Resolver

A cyan rounded rectangle with a drop shadow containing the text "URL Resolver".

- Reads anchors file
- Converts relative URLs into absolute URLs
- Converts absolute URLs into docIDs
- Inserts anchor text into forward index along with docID that anchor points to
- Generates links database containing pairs of docIDs
 - Used to compute PageRanks for all documents

Sorter

A cyan rounded rectangle with a drop shadow containing the text "Sorter".

- Processes forward index (docID sorted) to generate inverted index (wordID sorted)
- Short barrel - inverted index of title and anchor hits
- Full barrel - inverted index of full body text
- Includes document position offsets for each wordID into the inverted index (for proximity matching)

PageRank

PageRank

- Adds an *objective* measure of citation importance to backlinks that reference a site
- Can be described as model of user behavior
 - Assume “random surfer” given a web page at random and keeps clicking on links
 - User never hits back and eventually gets bored and requests another random page
 - Probability that the surfer will visit a page is PageRank
 - Probability that the surfer will request a random page is $(1-p)$

PageRank (cont.)

PageRank

- PageRank is defined as follows:

We assume page A has pages $T1...Tn$ which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1 (typically 0.85). Also, $C(A)$ is defined as the number of links going out of page A. The PageRank of a page A is given as follows:

$$PR(A) = (1-d) + d (PR(T1)/C(T1) + \dots + PR(Tn)/C(Tn))$$

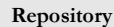
PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one.

PageRank (cont.)

A small cyan rounded rectangle with the word "PageRank" in black text.

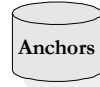
- Pages have a high PageRank if they have a lot of sites pointing to them
- Pages can also have a high PageRank if they are pointed to by a small number of sites with high PageRanks (e.g. Yahoo!)

Repository

A gray 3D cylinder representing a database or repository, with the word "Repository" written on its side.

- Compressed size 53.5 GB
- Full HTML of every web page
- Pages compressed using zlib compression
- Format:
 - docID
 - document length
 - document URL
 - document contents

Anchors



- Size 6.6 GB
- 24 million pages crawled generated over 259 million anchors
- Often more accurate description of page than page itself
- Anchors exist for pages that cannot be indexed by text-based search engine
- Potential problem of referencing sites that do not exist

Links



- 3.9 GB
- Database consisting of pairs of docIDs
- Used to compute PageRanks of all documents

Document Index

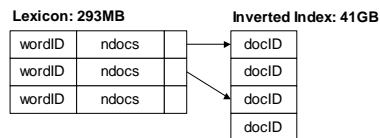


- Size 9.7 GB
- ISAM (Indexed sequential access mode) index ordered by docID
- Each entry contains:
 - Document status
 - Document checksum
 - Various statistics
- If document has been crawled (URL and title)
- Otherwise points to URL list

Lexicon



- Fits inside main memory
- 14 million words

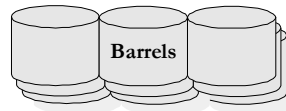


Barrels



- Size: Short Index, 4.1 GB, Full Index 37.1 GB
- Forward index (docID sorted) is in-place sorted to create the inverted index (wordID sorted)
- Search engine will look for hits in the short barrel (titles and anchors) before looking in the full barrel.

Barrels (cont.)



- Index formats:

Forward Index: 43GB

docID	wordID	numhits	hit	hit	hit	hit
	wordID	numhits	hit	hit	hit	hit
	null wordID					
docID	wordID	numhits	hit	hit	hit	hit
	wordID	numhits	hit	hit	hit	hit
	wordID	numhits	hit	hit	hit	hit
	null wordID					

← 24 bits → ← 8 bits →

Lexicon: 293MB

wordID	ndocs
wordID	ndocs
wordID	ndocs

Inverted Index: 41GB

docID	numhits	hit	hit	hit	hit	hit
docID	numhits	hit	hit	hit		
docID	numhits	hit	hit	hit	hit	hit
docID	numhits	hit	hit			

← 27 bits → ← 5 bits →

Google Searching

Searcher

- Google maintains much more information about web documents than typical search engines.
- Every hitlist includes position, font, and capitalization information. Additionally, hits from anchor text and the PageRank of the document are factored in.
- No particular factor can have too much influence.

Google Searching (cont.)

Searcher

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.

Ranking, Single Word Query

- Look at that the document hit list for the word
- Consider each hit to be one of type: {title, anchor, URL, plain text large font, plain text small font, ...}, each of which has its own type-weight
- Create vector of type-weights indexed by type
- Count number of hits of each type in the hit list, and convert each count into a count-weight.
- Count weight is normalized, linear at first, tapering off
- Weight score equals dot product of the vector of count-weights with vector of type-weights
- Weight score is combined with PageRank to give a final rank to the document.

Ranking, Multi-word query

- Scan multiple hit lists at once so that hits occurring close together in a document are weighted higher
- For every matched set of hits, compute proximity
- Classify each proximity into one of 10 categories ranging from “phrase match” to "not even close"
- Compute counts for not only for every type of hit, but also for every type and proximity pair
- Every type and proximity pair has a type-prox-weight.
- Counts are converted into count-weights
- Take dot product of the count-weights and type-prox-weights to compute a weight score.
- Weight score is combined with PageRank to give a final rank to the document.