The Nutch Open-Source Search Engine

CSE 454

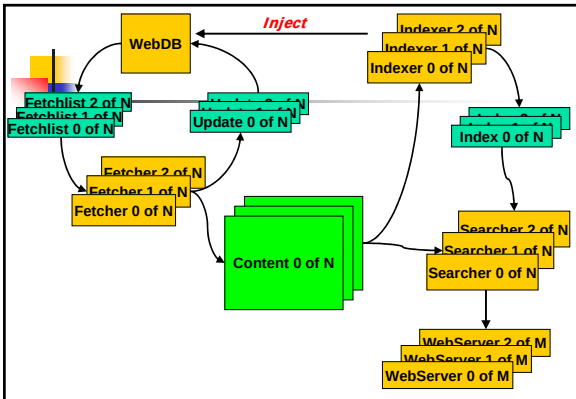
Slides by Michael J. Cafarella

Meta-details

- Built to encourage public search work
  - Open-source, w/pluggable modules
  - Cheap to run, both machines & admins
- Goal: Search more pages, with better quality, than any other engine
  - Pretty good ranking
  - Currently can do ~ 200M pages
    - << 8 Billion, but not too shabby

Outline

- Nutch design
  - Link database, fetcher, indexer, etc...
- Supporting parts
  - Distributed filesystem, job control



Moving Parts

- Acquisition cycle
  - WebDB
  - Fetcher
- Index generation
  - Indexing
  - Link analysis (maybe)
- Serving results

WebDB

- Contains info on all pages, links
  - URL, last download, # failures, link score, content hash, ref counting
  - Source hash, target URL
- Must always be consistent
- Designed to minimize disk seeks
  - 19ms seek time x 200m new pages/mo = ~44 days of disk seeks!

Fetcher

- Fetcher is very stupid. Not a "crawler"
  - Divide "to-fetch list" into *k* pieces,
  - One for each fetcher machine
- URLs for one domain go to same list, otherwise random
  - "Politeness" w/o inter-fetcher protocols
  - Can observe robots.txt similarly
  - Better DNS, robots caching
  - Easy parallelism
- Two outputs: pages, WebDB edits

WebDB/Fetcher Updates

WebDB
URL: http://www.cs.washington.edu/index.html
LastUpdated: 3/22/05
ContentHash: MD5_sdflkjweroiwelksd
URL: http://www.cnn.com/index.html
LastUpdated: Today!
ContentHash: MD5_balboglerropewolefbag
URL: http://www.yahoo.com/index.html
LastUpdated: 11/7/05
ContentHash: MD5_toewkekqmekkalekaa
URL: http://www.yahoo.com/index.html
LastUpdated: Today!
ContentHash: MD5_toewkekqmekkalekaa

Fetcher edits
Edit: DOWNLOAD_CONTENT
URL: http://www.yahoo/index.html
ContentHash: MD5_toewkekqmekkalekaa
Edit: DOWNLOAD_CONTENT
URL: http://www.cnn.com/index.html
ContentHash: MD5_balboglerropewolefbag
Edit: NEW_LINK
URL: http://www.flickr.com/index.html
ContentHash: None

4. Replace database with new database

Indexing

- Iterate through all *k* page sets in parallel, constructing inverted index
- Creates a "searchable document" of:
  - URL text
  - Content text
  - Incoming anchor text
- Other content types might have different document fields
  - Eg, email has sender/receiver
  - Any searchable field end-user will want
- Uses Lucene text indexer

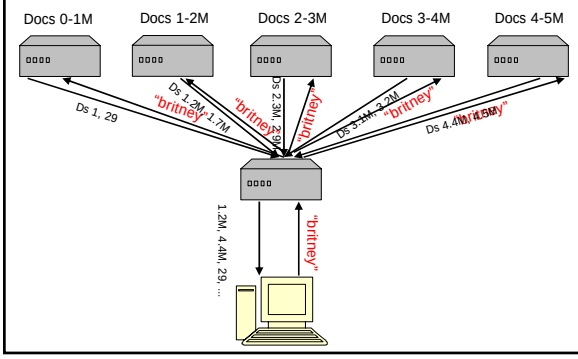
Link analysis

- A page's relevance depends on both intrinsic and extrinsic factors
  - Intrinsic: page title, URL, text
  - Extrinsic: anchor text, **link graph**
- E.g., PageRank, HITS, simple in-degree
- Sexy, but importance generally overstated

Link analysis (2)

- Nutch performs analysis in WebDB
  - Emit a score for each known page
  - At index time, incorporates score into index
- Extremely time-consuming
  - Disk-consuming, too (because we want to use low-memory machines)
- 0.5 * log(# incoming links)

Query Processing



Administering Nutch

- Admin costs are ***critical***
  - It's a hassle when you have 25 machines
  - Google has maybe ~150k
- Files
  - WebDB content, working files
  - Fetchlists, fetched pages
  - Link analysis outputs, working files
  - Inverted indices
- Jobs
  - Emit fetchlists, fetch, update WebDB
  - Run link analysis
  - Build inverted indices

Administering Nutch (2)

- Admin sounds boring, but it's ***not!***
  - *Really!*
  - *Mike swears!*
- Large-file maintenance
  - Nutch Distributed File System
    - Ala Google File System (Ghemawat et al.)
- Job Control
  - Map/Reduce (Dean and Ghemawat)
  - To be discussed soon

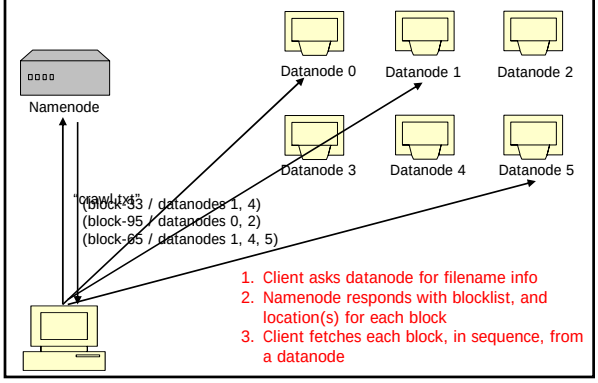
Nutch Distributed File System

- Similar, but not identical, to GFS
- Requirements are fairly strange
  - Extremely large files
  - Most files read once, from start to end
  - Low admin costs per GB
- Equally strange design
  - Write-once, with delete
  - Single file can exist across many machines
  - Wholly automatic failure recovery

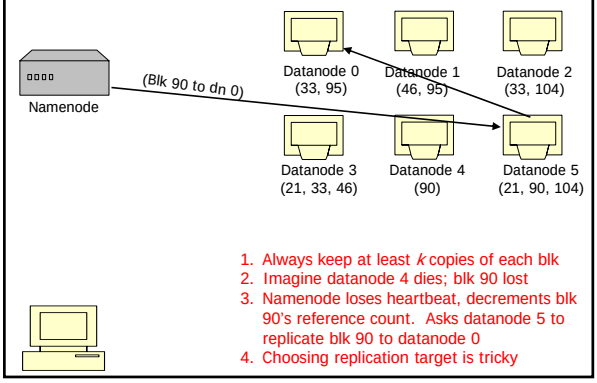
NDFS (2)

- Data divided into blocks
  - Blocks can be copied, replicated
- Datanodes hold and serve blocks
- Namenode holds metainfo
  - Filename → block list
  - Block → datanode-location
- Datanodes report to Namenode every few seconds,

NDFS File Read



NDFS Replication



Conclusion

- http://www.nutch.org/
  - Partial documentation
  - Source code
  - Developer discussion board
- “Lucene in Action” by Hatcher, Gospodnetic (you can borrow mine)
- Questions?