Nutch, and Search Engine History

Michael J. Cafarella

CSE 490H

October 21, 2008

Agenda

- Nutch in-depth
- A Technical History of Search Engines

Nutch

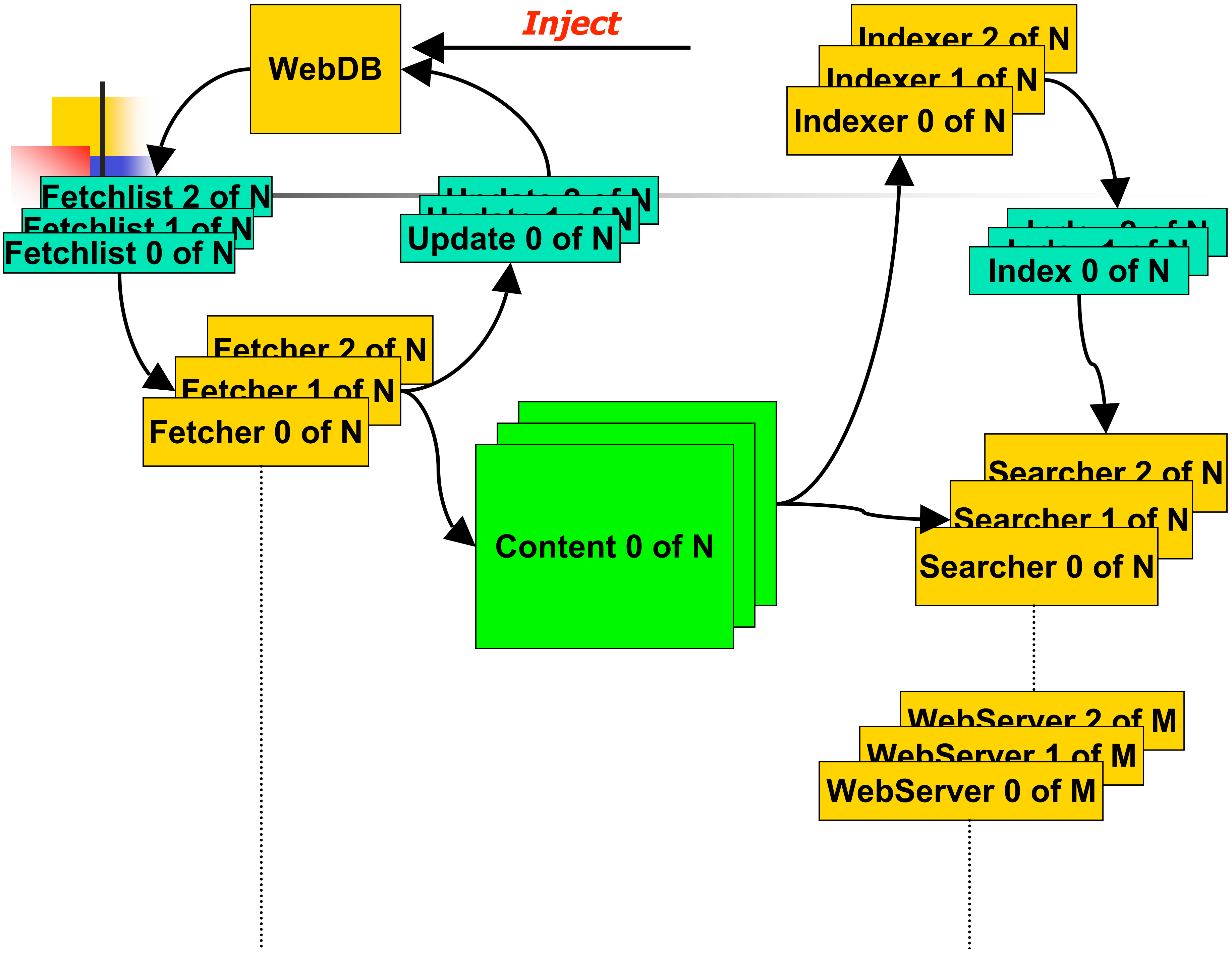
- Built to encourage public search work
  - Open-source, w/pluggable modules
  - Cheap to run, both machines & admins
- Search engine is usable, not great
  - Pretty good ranking (last rigorous test several years ago showed roughly Inktomi-level quality)
  - Has done ~ 200M pages, more possible
- Hadoop is a spinoff

Timeline

- Fall, 2002 - Nutch started with ~2 people
- Summer, 2003 - 50M pages demo’ed
- Fall, 2003 - Google File System paper
- Summer, 2004 - Distributed indexing, started work on GFS clone
- Fall, 2004 - MapReduce paper
- 2005 - Started work on MapReduce. Massive Nutch rewrite, to move to GFS & MapReduce framework
- 2006 - Hadoop spun out, Nutch work slows
- 2007 - Widespread Hadoop adoption

Outline

- Nutch design
  - Link database, fetcher, indexer, etc…
- Hadoop support
  - Distributed filesystem, job control



Inject

WebDB

Fetchlist 2 of N

Fetchlist 1 of N

Fetchlist 0 of N

Update 2 of N

Update 1 of N

Update 0 of N

Fetcher 2 of N

Fetcher 1 of N

Fetcher 0 of N

Content 0 of N

Indexer 2 of N

Indexer 1 of N

Indexer 0 of N

Index 2 of N

Index 1 of N

Index 0 of N

Searcher 2 of N

Searcher 1 of N

Searcher 0 of N

WebServer 2 of M

WebServer 1 of M

WebServer 0 of M

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!
- Single-disk WebDB was huge headache

Fetcher

- Fetcher is very stupid. Not a “crawler”
- Pre-MapRed: 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

URL: http://www.about.com/index.html
LastUpdated: 3/22/05
ContentHash: MD5_sdflkjweroiwelksd

URL: http://www.cnn.com/index.html
LastUpdated: Never LastUpdated: Today!
ContentHash: None ContentHash: MD5_balboglerropewolefbag

URL: http://www.flickr.com/index.html URL: http://www.yahoo/index.html
LastUpdated: Never LastUpdated: 4/07/05
ContentHash: None ContentHash: MD5_toewkekqmekkalekaa

URL: http://www.yahoo.com/index.html
LastUpdated: Today!
ContentHash: MD5_toewkekqmekkalekaa

WebDB

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

Fetcher edits

1. Write down fetcher edits

2. Sort edits (externally, if necessary)

3. Read streams in parallel, emitting new database

4. Repeat for other tables

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 a 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**
- PageRank is most famous of many
- Others include:
  - HITS
  - OPIC
  - Simple incoming link count
- Link analysis is sexy, but importance generally overstated

Link analysis (2)

- Nutch performs analysis in WebDB
  - Emit a score for each known page
  - At index time, incorporate score into inverted index
- Extremely time-consuming
  - In our case, disk-consuming, too (because we want to use low-memory machines)
- Fast and easy:
  - 0.5 * log(# incoming links)

Administering Nutch

- Admin costs are critical
  - It’s a hassle when you have 25 machines
  - Google has >100k, probably more
- 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
  - I swear
- Large-file maintenance
  - Google File System (Ghemawat, Gobioff, Leung)
  - Nutch Distributed File System
- Job Control
  - Map/Reduce (Dean and Ghemawat)
  - Pig (Yahoo Research)
- Data Storage (BigTable)

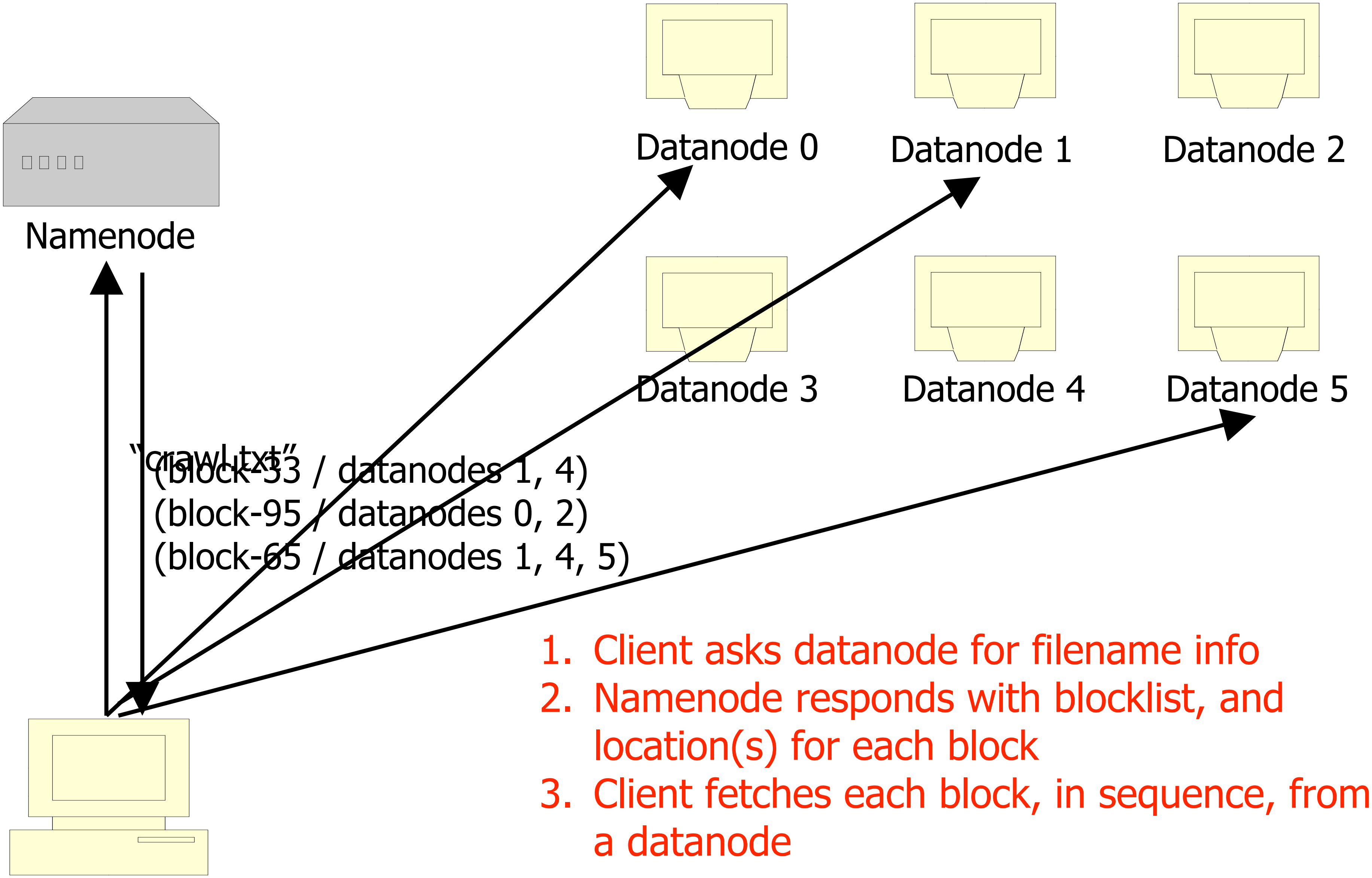
Nutch Distributed File System

- 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 in to namenode every few seconds

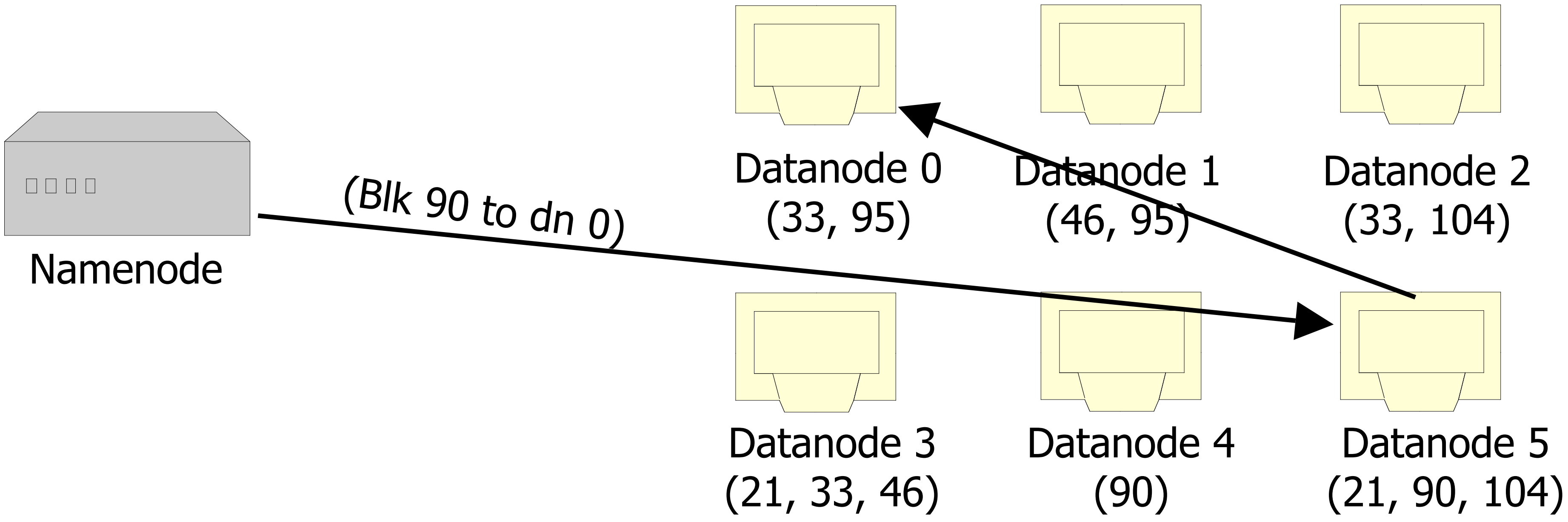
NDFS File Read

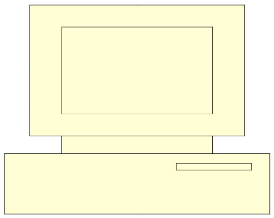


(block-33 / datanodes 1, 4)
(block-95 / datanodes 0, 2)
(block-65 / datanodes 1, 4, 5)

1. Client asks datanode for filename info
2. Namenode responds with blocklist, and location(s) for each block
3. Client fetches each block, in sequence, from a datanode

NDFS Replication





1. Always keep at least *k* copies of each blk
2. Imagine datanode 4 dies; blk 90 lost
3. Namenode loses heartbeat, decrements blk 90’s reference count. Asks datanode 5 to replicate blk 90 to datanode 0
4. Choosing replication target is tricky

Nutch & Hadoop

- NDFS stores the crawl and indexes
- MapReduce for indexing, parsing, WebDB construction, even fetching
  - Broke previous 200M/mo limit
  - Index-serving?
- Required massive rewrite of almost every Nutch component

Nutch Conclusion

- http://www.nutch.org/
  - Partial documentation
  - Source code
  - Developer discussion board
- Nutch has been only moderately successful, but led to Hadoop
- “Lucene in Action” by Hatcher, Gospodnetic is a useful resource

Search: A Technical History

- Search engines have been around a lot longer than you think
- Almost all of them are dead and gone, but their ideas live on
- Search existed before the Web, though it was a very different beast

Primordial Era: 1960s-1994

- Electronic content was rare and expensive
- Only large organizations with huge well-curated archives (libraries, govts) had any need for search
- CPU & storage were expensive, networked systems very rare
- Most systems were small, searched only metadata (like card catalogs)

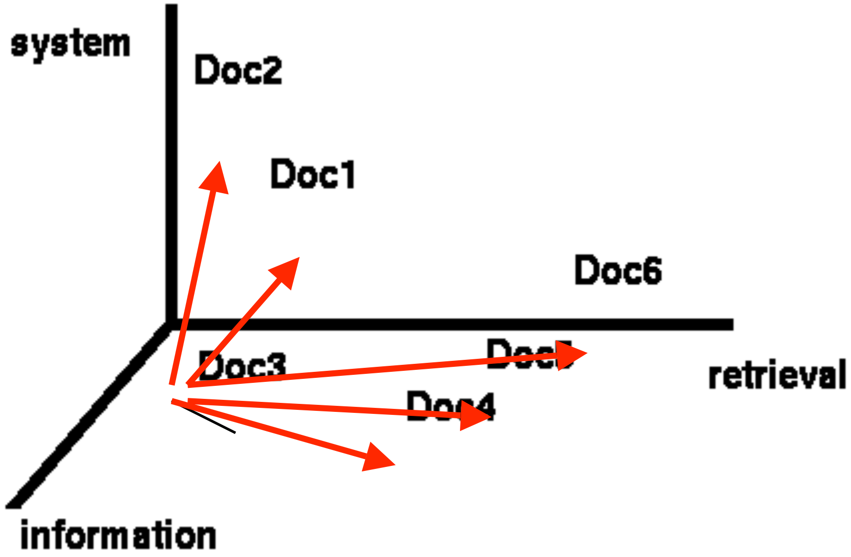
Primordial Era (2)

- Two important technical contributions
  - Inverted index
  - Tf/idf & vector document model
- Document ranking was not a huge problem
  - Relatively few documents
  - Clean metadata
  - Boolean operators commonplace

Inverted Index: why bother?

- Disk access: 1-10ms
  - Depends on seek distance, published average is 5ms
  - Thus perform 200 seeks / sec
  - (And we are ignoring rotation and transfer times)
- Clock cycle: 2 GHz
  - Typically *completes* 2 instructions / cycle
    - ~10 cycles / instruction, but pipelining & parallel execution
  - Thus: 4 billion instructions / sec
- Disk is *20 Million* times slower
- Inverted index allows us to read all of the docs for a single search term, usually with a single seek.
- # seeks grows with # terms, not # documents.

Tf/idf: Vector Model

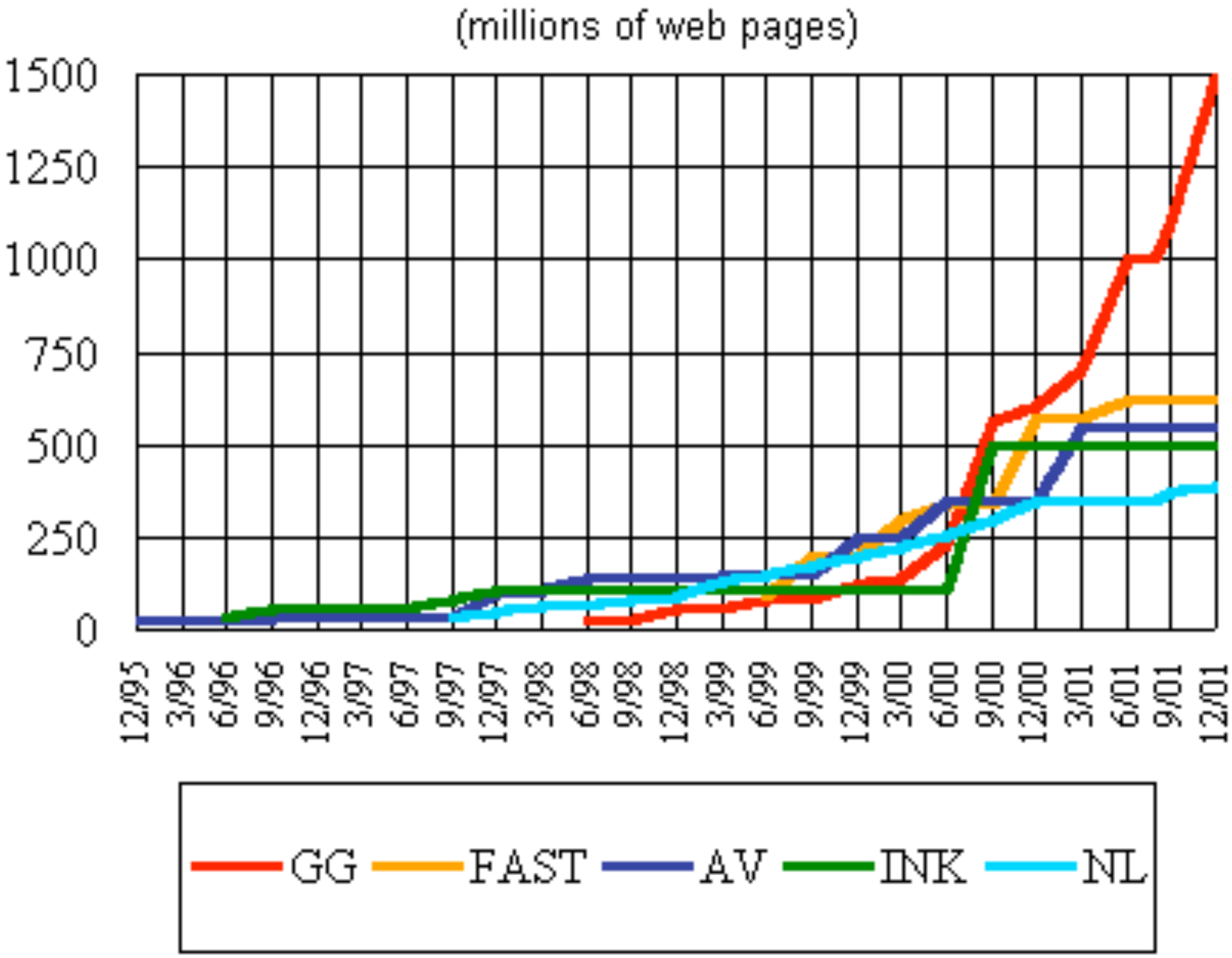


- Tf = term frequency, idf = inverse document frequency; tf/idf for a term places it in N-dim space
- Documents that are “close together” in space are similar in meaning.

The Web (1994-)

- The popularization of the Web in the 1990s led to a crazy explosion of search engine companies
- Web search was a vastly different problem compared to previous systems
  - Content was cheap but messy
  - Storage was becoming cheap
  - Finding a document became harder
  - Users were much less sophisticated

Search Engine Size over Time



Number of indexed pages, self-reported

Information from searchenginewatch.com

Search Engine Storage Costs

- Figure 10kb to index one Web page plus a compressed cached copy
- In 2008, 1GB costs ~0.15
  - 100k docs per gig, so $0.0000015/doc
  - 50M docs costs $75.00
- In 1990, 1GB costs $1000.00
  - 100k docs per gig, so $0.01/doc
  - 50M docs costs $500k
  - Just about within reach for startup search companies

WebCrawler

- Created in 1994 by a UW student!
- Notable features:
  - First dynamic crawler (rather than using hand-curated corpus)
- Fate:
  - Bought by AOL, then Excite, then InfoSpace
  - Now a meta-engine, serving results from elsewhere

Excite (aka Architext)

- Created in 1994 by Stanford ugrads
- Notable features:
  - Full-text indexing for Web pages
  - “Related search” suggestions
  - Famous in mid-90s for consuming tons of expensive high-end Sun machines
- Fate:
  - Went public, bought many other companies, merged with @Home, collapsed in bankruptcy, then sold for parts

Infoseek

- Created in 1994
- Notable features:
  - Very fancy query language (booleans, NEAR, etc)
  - Performed some linguistic analysis, including stemming. Gave stemming a bad name for a decade.
- Fate:
  - Bought by Disney in 1998

Inktomi

- Created in 1996 by UCB grad student
- Notable features:
  - Distributed commodity-box infrastructure
  - Resold its search engine to other destination sites (Hotbot, Yahoo, others)
  - Search was just one of several products (others were caches and video serving)
- Fate:
  - Went public, stock collapsed in crash, sold to Yahoo in 2002

AltaVista

- Created in 1995 as a DEC research project
- Notable Features:
  - Originally meant to demo new 64-bit Alpha processor: high speed & huge address space
  - First really high-quality multithreaded crawler: 30m pages at launch!
  - Recognized that page ranking was an issue, but used awful solution: URL length
- Fate:
  - Compaq bought DEC, then sold AV to CMGI, which sold AV to Overture, which was then bought by Yahoo

Google

- Founded in 1998. Have you heard of it?
- Major feature was PageRank (Page, 1998)
  - Largely solved page-ranking problem faced by AltaVista
  - First major commercial deployment of link-based methods
  - Really miraculous when compared to other methods at the time
- However, link-based methods were common in academia
  - Authoritative Sources in a Hyperlinked Environment, Kleinberg. JACM, 1999.
  - “Silk from a sow’s ear”, Pirolli, Pitkow, Rao. CHI, 1996.

Google (2)

- PageRank is its best-known contribution, but Google was helped by its predecessors:
  - Full-text indexing, like Excite
  - An aggressive large-scale crawler, like WebCrawler and AltaVista
  - Distributed processing from Inktomi
- The last interesting Web search engine?
  - Probably. Previous search engines got a ton of traffic. They just didn’t have ad revenue
  - The period 1994-1998 was very unusual, made possible by the Web’s split between search and content ownership