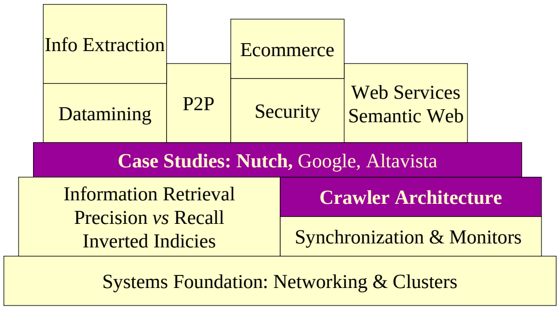
Crawlers: Nutch

CSE 454

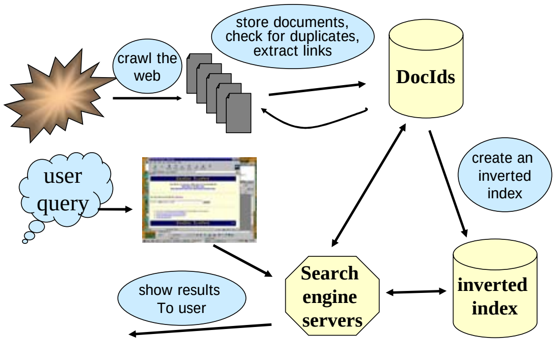
Administrivia

- **Groups Formed**
- **Architecture Documents under Review**
- **Group Meetings**

Course Overview


Info Extraction
Datamining
P2P
Ecommerce
Security
Web Services
Semantic Web
Case Studies: Nutch, Google, Altavista
Information Retrieval
Precision *vs* Recall
Inverted Indicies
Crawler Architecture
Synchronization & Monitors
Systems Foundation: Networking & Clusters

Standard Web Search Engine Architecture


crawl the web
store documents, check for duplicates, extract links
DocIds
create an inverted index
user query
Search engine servers
inverted index
show results To user

Slide adapted from Marty Hearst / UC Berkeley]

Issues

- **Crawling**
- **Search**
- **Presentation**

Crawling Issues

- **Storage efficiency**
- **Search strategy**
  - Where to start
  - Link ordering
  - Circularities
  - Duplicates
  - Checking for changes
- **Politeness**
  - Forbidden zones: robots.txt
  - CGI & scripts
  - Load on remote servers
  - Bandwidth (download what need)
- **Parsing pages for links**
- **Scalability**

Searching Issues

- **Scalability (how measure speed?)**
- **Ranking**
- **Boolean queries**
- **Phrase search**
- **Nearness**
- **Substrings & stemming**
- **Stop words**
- **Multiple languages**
- **Spam, cloaking, …**
- **Multiple meanings for search words**
- **File types: images, audio, …**
- **Updating the index**

Thinking about Efficiency

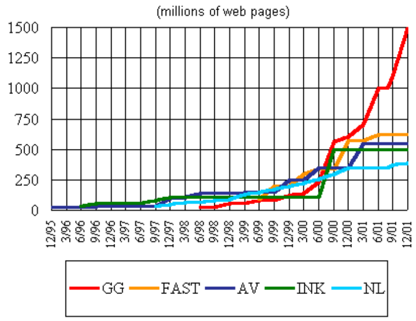
- **Disk access: 1-10ms**
  - Depends on seek distance, published average is 5ms
  - Thus perform 200 seeks / sec
  - (And we are ignoring rotation and transfer times)
- **Clock cycle: 2 GHz**
  - Typically *completes* 2 instructions / cycle
    - ~10 cycles / instruction, but pipelining & parallel execution
  - Thus: 4 billion instructions / sec
- **Disk is *20 Million* times slower !!!**

- **Store index in Oracle database?**
- **Store index using files and unix filesystem?**

Search Engine Architecture

- **Spider**
  - Crawls the web to find pages. Follows hyperlinks. Never stops
- **Indexer**
  - Produces data structures for fast searching of all words in the pages
- **Retriever**
  - Query interface
  - Database lookup to find hits
    - 300 million documents
    - 300 GB RAM, terabytes of disk
  - Ranking, summaries
- **Front End**

Search Engine Size over Time



Number of indexed pages, self-reported
Google: 50% of the web?

Crawlers (Spiders, Bots)

- **Retrieve web pages for indexing by search engines**
- **Start with an initial page P_0.**
- **Find URLs on P_0 and add them to a queue**
- **When done with P_0, pass it to an indexing program, get a page P_1 from the queue and repeat**
- **Can be specialized (e.g. only look for email addresses)**
- **Issues**
  - Which page to look at next? (keywords, recency, ?)
  - Avoid overloading a site
  - How deep within a site to go (drill-down)?
  - How frequently to visit pages?

Spiders

- **243 active spiders registered 1/01**
  - http://info.webcrawler.com/mak/projects/robots/active/html/index.html
- **Inktomi Slurp**
  - Standard search engine
- **Digimark**
  - Downloads just images, looking for watermarks

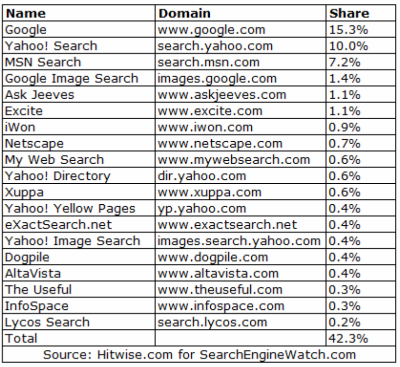
- **Adrelevance**
  - Looking for Ads.

Searches / Day

Google	250 M
Overture	167 M
Inktomi	80 M
LookSmart	45 M
FindWhat	33 M
AskJeeves	20 M
Altavista	18 M
FAST	12 M

From SearchEngineWatch 02/03

Hitwise: Search Engine Ratings



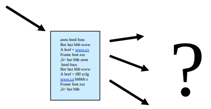
Name	Domain	Share
Google	www.google.com	15.3%
Yahoo! Search	search.yahoo.com	10.0%
MSN Search	search.msn.com	7.2%
Google Image Search	images.google.com	1.4%
Ask Jeeves	www.askjeeves.com	1.1%
Excite	www.excite.com	1.1%
iWon	www.iwon.com	0.9%
Netscape	www.netscape.com	0.7%
My Web Search	www.mywebsearch.com	0.6%
Yahoo! Directory	dir.yahoo.com	0.6%
Xuppa	www.xuppa.com	0.6%
Yahoo! Yellow Pages	yp.yahoo.com	0.4%
eXactSearch.net	www.exactsearch.net	0.4%
Yahoo! Image Search	images.search.yahoo.com	0.4%
Dogpile	www.dogpile.com	0.4%
AltaVista	www.altavista.com	0.4%
The Useful	www.theuseful.com	0.3%
InfoSpace	www.infospace.com	0.3%
Lycos Search	search.lycos.com	0.2%
Total		42.3%

Source: Hitwise.com for SearchEngineWatch.com

5/04

Outgoing Links?

- Parse HTML…
- Looking for…what?



Which tags / attributes hold URLs?

Anchor tag: <a href="URL" … > … </a>

Option tag: <option value="URL"…> … </option>

Map: <area href="URL" …>

Frame: <frame src="URL" …>

Link to an image: <img src="URL" …>

Relative path vs. absolute path: <base href= …>

Robot Exclusion

- Person may not want certain pages indexed.
- Crawlers should obey Robot Exclusion Protocol.
  - But some don't
- Look for file **robots.txt** at highest directory level
  - If domain is www.ecom.cmu.edu, robots.txt goes in www.ecom.cmu.edu/robots.txt
- Specific document can be shielded from a crawler by adding the line:

<META NAME="ROBOTS" CONTENT="NOINDEX">

Robots Exclusion Protocol

- Format of robots.txt
  - Two fields. User-agent to specify a robot
  - Disallow to tell the agent what to ignore
- To exclude all robots from a server:

```
User-agent: *
Disallow: /
```

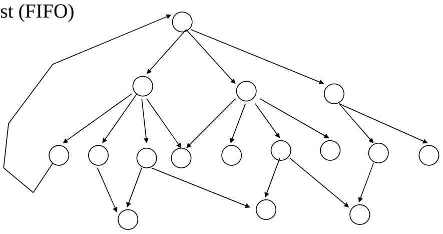
- To exclude one robot from two directories:

```
User-agent: WebCrawler
Disallow: /news/
Disallow: /tmp/
```

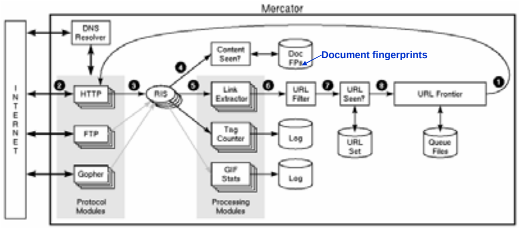
- View the robots.txt specification at http://info.webcrawler.com/mak/projects/robots/norobots.html

Web Crawling Strategy

- **Starting location(s)**
- **Traversal order**
  - Depth first (LIFO)
  - Breadth first (FIFO)
  - Or ???
- **Politeness**
- **Cycles?**
- **Coverage?**



Structure of Mercator Spider



1. Remove URL from queue
2. Simulate network protocols & REP
3. Read w/ RewindInputStream (RIS)
4. Has document been seen before? (checksums and fingerprints)
5. Extract links
6. Download new URL?
7. Has URL been seen before?
8. Add URL to frontier

URL Frontier (priority queue)

- **Most crawlers do breadth-first search from seeds.**
- **Politeness constraint: don’t hammer servers!**
  - Obvious implementation: “live host table”
  - Will it fit in memory?
  - Is this efficient?
- **Mercator’s politeness:**
  - One FIFO subqueue per thread.
  - Choose subqueue by hashing host’s name.
  - Dequeue first URL whose host has NO outstanding requests.

Fetching Pages

- **Need to support http, ftp, gopher, ...**
  - Extensible!
- **Need to fetch multiple pages at once.**
- **Need to cache as much as possible**
  - DNS
  - robots.txt
  - Documents themselves (for later processing)
- **Need to be defensive!**
  - Need to time out http connections.
  - Watch for “crawler traps” (e.g., infinite URL names.)
  - See section 5 of Mercator paper.
  - Use URL filter module
  - Checkpointing!

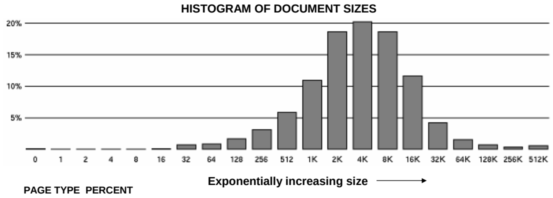
(A?) Synchronous I/O

- **Problem: network + host latency**
  - Want to GET multiple URLs at once.
- **Google**
  - Single-threaded crawler + asynchronous I/O
- **Mercator**
  - Multi-threaded crawler + synchronous I/O
  - Easier to code?

Duplicate Detection

- **URL-seen test: has this URL been seen before?**
  - To save space, store a hash
- **Content-seen test: different URL, same doc.**
  - Supress link extraction from mirrored pages.
- **What to save for each doc?**
  - 64 bit “document fingerprint”
  - Minimize number of disk reads upon retrieval.

Mercator Statistics



PAGE TYPE	PERCENT
text/html	69.2%
image/gif	17.9%
image/jpeg	8.1%
text/plain	1.5
pdf	0.9%
audio	0.4%
zip	0.4%
postscript	0.3%
other	1.4%

Advanced Crawling Issues

- **Limited resources**
  - Fetch most ***important*** pages first
- **Topic specific search engines**
  - Only care about pages which are ***relevant*** to topic

“Focused crawling”

- **Minimize stale pages**
  - Efficient re-fetch to keep index timely
  - How track the rate of change for pages?

Focused Crawling

- **Priority queue instead of FIFO.**
- **How to determine priority?**
  - Similarity of page to driving query
    - Use traditional IR measures
  - Backlink
    - How many links point to this page?
  - PageRank (Google)
    - Some links to this page count more than others
  - Forward link of a page
  - Location Heuristics
    - E.g., Is site in .edu?
    - E.g., Does URL contain ‘home’ in it?
  - Linear combination of above