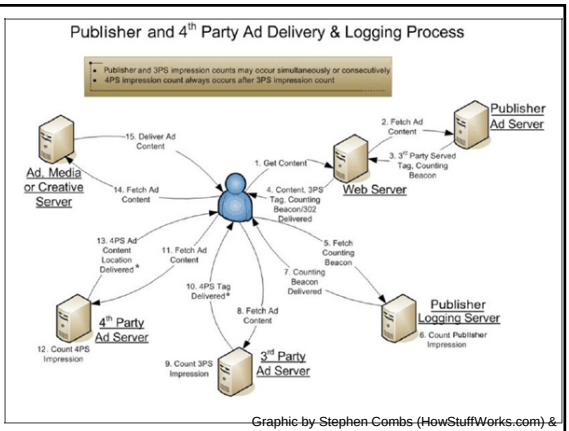
Crawling HTML

Class Overview

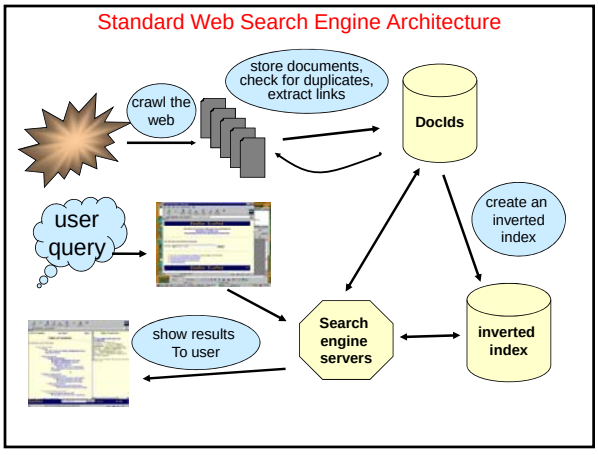
- Other Cool Stuff
- Query processing
- Content Analysis
- Indexing
- Crawling
- Document Layer
- Network Layer

Today

- Crawlers
- Server Architecture



Graphic by Stephen Combs (HowStuffWorks.com) & Kari Meoller(Turner Broadcasting)



CRAWLERS…

Danger Will Robinson!!

- Consequences of a bug

Max 6 hits/server/minute
plus….

http://www.cs.washington.edu/lab/policies/crawlers.html

Open-Source Crawlers

- GNU Wget
  - Utility for downloading files from the Web.
  - Fine if you just need to fetch files from 2-3 sites.
- Heritix
  - Open-source, extensible, Web-scale crawler
  - Easy to get running.
  - Web-based UI
- Nutch
  - Featureful, industrial strength, Web search package.
  - Includes Lucene information retrieval part
    - TF/IDF and other document ranking
    - Optimized, inverted-index data store
  - You get complete control thru easy programming.

Search Engine Architecture

- Crawler (Spider)
  - Searches 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

Thinking about Efficiency

- Clock cycle: 2 GHz
  - Typically *completes* 2 instructions / cycle
    - ~10 cycles / instruction, but pipelining & parallel execution
  - Thus: 4 billion instructions / sec
- 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)
- Disk is *20 Million* times slower !!!
- Store index in Oracle database?
- Store index using files and unix filesystem?

10/12/2010 6:07 PM 10

Spiders = Crawlers

- 1000s of spiders
- Various purposes:
  - Search engines
  - Digital rights management
  - Advertising
  - Spam
  - Link checking – site validation

Spiders (Crawlers, Bots)

- Queue := initial page URL_0
- Do forever
  - Dequeue URL
  - Fetch P
  - Parse P for more URLs; add them to queue
  - Pass P to (specialized?) indexing program
- Issues…
  - Which page to look at next?
    - keywords, recency, focus, ???
  - Avoid overloading a site
  - How deep within a site to go?
  - How frequently to visit pages?
  - Traps!

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
- Malicious servers: SEOs

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:

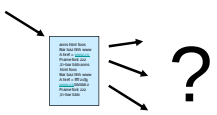
  User-agent: WebCrawler
  Disallow: /news/
  Disallow: /tmp/
- View the robots.txt specification at
  http://info.webcrawler.com/mak/projects/robots/norobots.html

Danger, Danger

- **Ensure that your crawler obeys robots.txt**
- **Don't make any of these typical mistakes:**
  - **Provide contact info in user-agent field.**
  - **Monitor the email address**
  - **Notify the CS Lab Staff**
  - **Honor all Do Not Scan requests**
  - **Post any "stop-scanning" requests**
  - **"The scanee is *always* right."**
  - **Max 6 hits/server/minute**

Outgoing Links?

- Parse HTML…
- Looking for…what?



?

Which tags / attributes hold URLs?

Anchor tag: …

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= …>

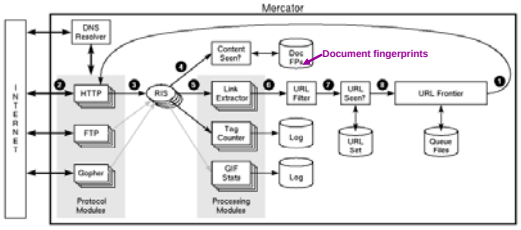
Bonus problem: Javascript

In our favor: Search Engine Optimization

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!

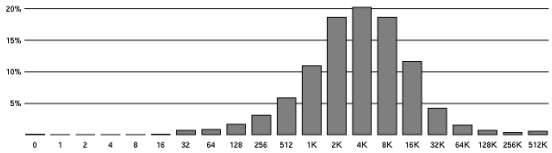
Duplicate Detection

- URL-seen test: has 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.

Nutch: A simple architecture

- Seed set
- Crawl
- Remove duplicates
- Extract URLs (minus those we've been to)
  - new frontier
- Crawl again
- Can do this with Map/Reduce architecture

Mercator Statistics



Exponentially increasing size

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%

Outdated

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

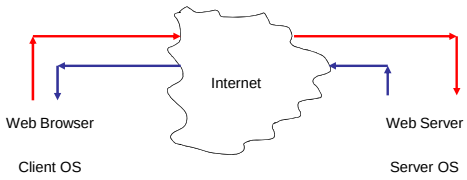
- Priority queue instead of FIFO.
- How to determine priority?
  - Similarity of page to driving query
    - Use traditional IR measures
    - Exploration / exploitation problem
  - Backlink
    - How many links point to this page?
  - PageRank (Google)
    - Some links to this page count more than others
  - 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

Outline

- Search Engine Overview
- HTTP
- Crawlers
- Server Architecture

Server Architecture

Connecting on the WWW



Client-Side View

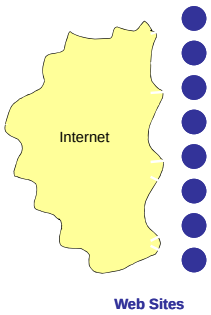
Content rendering engine
- Tags, positioning, movement

Scripting language interpreter
- Document object model
- Events
- Programming language itself

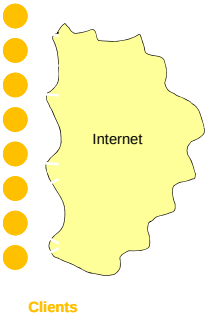
Link to custom Java VM

Security access mechanisms

Plugin architecture + plugins



Server-Side View



- Database-driven content
- Lots of Users
  - Scalability
  - Load balancing
- Often implemented with cluster of PCs
- 24x7 Reliability
- Transparent upgrades

Trade-offs in Client/Server Arch.

- Compute on clients?
  - – Complexity: Many different browsers
    - {Firefox, IE, Safari, …} × Version × OS
- Compute on servers?
  - – Peak load, reliability, capital investment.
  - + Access anywhere, anytime, any device
  - + Groupware support (shared calendar, …)
  - + Lower overall cost (utilization & debugging)
  - + Simpler to update service

Dynamic Content

- We want to do more via an http request
  - – E.g. we'd like to invoke code to run on the server.
- Initial solution: Common Gateway Interface (CGI) programs.
- Example: web page contains form that needs to be processed on server.

CGI Code

- CGI scripts can be in any language.
- A new process is started (and terminated) with each script invocation (overhead!).
- Improvement I:
  - – Run some code on the client's machine
  - – E.g., catch missing fields in the form.
- Improvement II:
  - – Server APIs (but these are server-specific).

Java Servlets

- Servlets : applets that run on the server.
  - – Java VM stays, servlets run as threads.
- Accept data from client + perform computation
- Platform-independent alternative to CGI.
- Can handle multiple requests concurrently
  - – Synchronize requests - use for online conferencing
- Can forward requests to other servers
  - – Use for load balancing

Java Server Pages (JSP)
Active Server Pages (ASP)

- Allows mixing static HTML w/ dynamically generated content
- JSP is more convenient than servlets for the above purpose
- More recently PHP & Ruby on Rails

```
<html>

<head>
<title>Example #3</title>
</head>
<? print(Date("m/j/y")); ?>

<body>
</body>
</html>
```

AJAX

- Getting the browser to behave like your applications (caveat: Asynchronous)
- Client → Rendering library (Javascript)
  - Widgets
- Talks to Server (XML)
- How do we keep state?
- Over the wire protocol: SOAP/XML-RPC/etc.

Interlude: HTML 5

Why HTML 5?

'The websites of today are built with languages largely conceived during the mid to late1990's, when the web was still in its infancy.'*

* Work on HTML 4 started in early 1997
CSS 2 was published in 1998

Slide from David Penny, EMCDDA 11/09

The website circa 1998

- Simple layout
- No frills design
- Text, text, text

Slide from David Penny, EMCDDA 11/09

The website circa 2009

- Complex layout
- Fancy designs
- User-interactivity

The modern web page is sometimes like a book,
sometimes like an application,
sometimes like an extension of your TV.
Current web languages were never designed to do this.

Slide from David Penny, EMCDDA 11/09

HTML 5 & CSS 3

HTML 5

- Specifically designed for web applications
- Nice to search engines and screen readers
- *HTML 5 will update HTML 4.01, DOM Level 2*

CSS level 3

- Will make it easier to do complex designs
- Will look the same across all browsers
- *CSS 3 will update CSS level 2 (CSS 2.1)*

Slide from David Penny, EMCDDA 11/09

HTML 5: today's markup

- Today, if we wanted to markup this page we would use a lot of tags, and classes.
- Semantic value of and 'class' = 0
-
- Can lead to 'divitis' and 'classitis'.

Slide from David Penny, EMCDDA 11/09

HTML 5: new tags to the rescue

- Hello ,<header>, <nav>, <article>, <section>, and other new tags.
- It's good for search engines, screen readers, information architects, and the web in general.

Slide from David Penny, EMCDDA 11/09

HTML 5: at last, video + audio

- Currently Video and audio handled by plugins (Flash, ReatTime, etc.)
- New **<video>** and **<audio>** and associated APIs tags will be used as **<img>** tag is today
- Browsers will need to define how video and audio should be played (controls, interface, etc.)

Slide from David Penny, EMCDDA 11/09

HTML 5: Web applications 1.0

- Web applications a huge part of HTML 5.
- Some APIs include:
  - drag and drop,
  - canvas (drawing),
  - offline storage,
  - geo-location,

Slide from David Penny, EMCDDA 11/09

HTML 5: Form handling

- **required** attribute:
  - browser checks for you that the data has been entered
- **email** input type:
  - a valid email must be entered
- **url** input type:
  - requires a valid web address

Slide from David Penny, EMCDDA 11/09

Roadmap

- First W3C Working Draft in October 2007.
- Last Call Working Draft in October 2009.
- **Candidate Recommendation in 2012.**
- First and second draft of test suite in 2012, 2015.
- Reissued Last Call Working Draft in 2020.
- Proposed Recommendation in 2022 (!)
- **Current browsers have already started implementing HTML 5.**

Note: today's candidate recommendation status = yesterday's recommendation status

Slide from David Penny, EMCDDA 11/09

CSS-3: round corners

- *border-radius* (or variant depending on browser) is used to make rounded corners
- Example:
  `border-radius: 3px`
- The bigger the value or the radius, the more curvy and larger are the rounded corners
- Much simpler than using CSS 2 (no background images etc. needed)

Slide from David Penny, EMCDDA 11/09

CSS-3: Multi-column layout

- Allows you to split text newspaper-like across multiple columns
- Express in terms of number of columns or width.
- Example 1:
  ```
  column-width: 45%;
  column-gap 5%;
  ```
- Example 2:
  ```
  column-count: 3;
  ```

Slide from David Penny, EMCDDA 11/09

CSS-3: Other new stuff

- Multiple backgrounds
- Border images
- Transitions
- New selectors:

'... the power to describe Web 2.0 designs in CSS is insignificant compared with the power to select every third table row starting with the fifth one.'
Eric Meyer

Slide from David Penny, EMCDDA 11/09

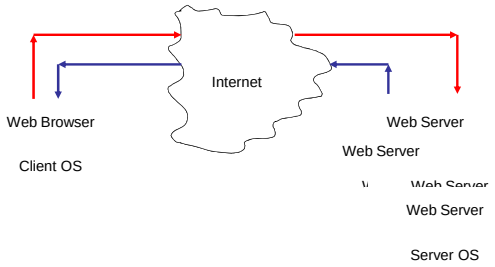
CSS 3 timeline

- Unlike CSS 2, CSS 3 consists of modules
- Each module is recommended separately
- Several modules are already considered stable and will probably not change in the future
- Many are already implemented in current browsers
- **www.w3.org/Style/CSS/current-work** gives the state of each module

Slide from David Penny, EMCDDA 11/09

Server Architecture

Connecting on the WWW



Tiered Architectures

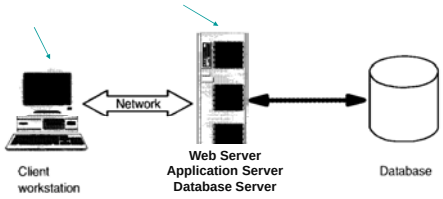
1-tier = dumb terminal → smart server.

2-tier = client/server.

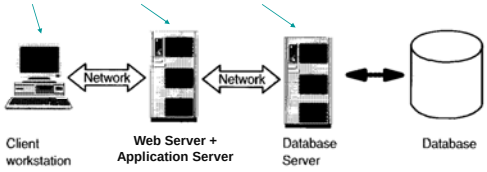
3-tier = client/application server/database.

Why decompose the server?

Two-Tier Architecture

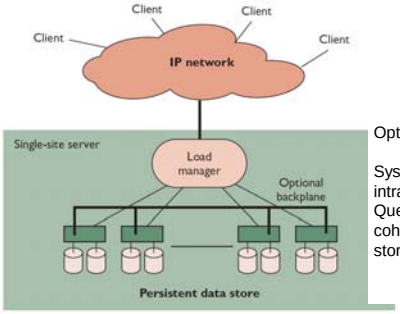


Three-Tier Architecture



Getting to 'Giant Scale'

- Only real option is cluster computing



Optional Backplane:

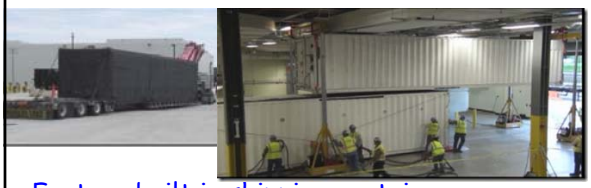
System-wide network for intra-server traffic: Query redirect, coherence traffic for store, updates, …

Microsoft Server Farm

Quincy, WA

9th largest in US (as of May 2010)

Containerized Data Centers



- Factory built in shipping container
- Trucked to loc; forklift stacks in warehouse
- Connected to:
  - chilled water supply,
  - fiber-optic connection,
  - electrical plugs
- Self-provisioning +self-managed.

Inside the Container

- Extreme symmetry
- Internal disks
- No monitors
- No visible cables
- No people!

- Offsite management
- Contracts limit
  - Δ Power
  - Δ Temperature

From: Brewer *Lessons from Giant-Scale Services*
Image: Microsoft Chicago data center

High Availability

- Essential Objective
- Phone network, railways, water system
- Challenges
  - Component failures
  - Constantly evolving features
  - Unpredictable growth

From: Brewer *Lessons from Giant-Scale Services*

Architecture

- What do faults impact? Yield? Harvest?
- Replicated systems
  Faults → reduced capacity (hence, yield @ high util)
- Partitioned systems
  Faults → reduced harvest
  Capacity (queries / sec) unchanged

- DQ Principle ∃ physical bottleneck
  Data/Query × Queries/Sec = Constant

From: Brewer *Lessons from Giant-Scale Services*

Graceful Degradation

- Too expensive to avoid saturation
- Peak/average ratio
  - 1.6x - 6x or more
  - Moviefone: 10x capacity for Phantom Menace
    - Not enough…
- Dependent faults (temperature, power)
  - Overall DQ drops ***way*** down

- Cutting harvest by 2 doubles capacity…

Admission Control (AC) Techniques

- Cost-Based AC
  - Denying an expensive query allows 2 cheap ones
  - Inktomi
- Priority-Based (Value-Based) AC
  - Stock trades *vs.* quotes
  - Datek
- Reduced Data Freshness