Collection Building on the Web

Basic Algorithm

Initialize URL queue
While more URLs
  If URL is not a duplicate
    Get document with URL
    [Add to database]
    Extract URLs, add to queue

Collection Goals

- Completeness
- Topicality
- Quality
- Freshness
- Low duplication

Collection decisions

- Which pages to download
  - Interest-driven
  - Popularity-driven
  - Location-driven
- How should the crawler refresh pages

  What are we trying to maximize?
- How do we avoid over-taxing a site when it's crawled?

What to collect

- What are your users trying to find?
  - Pages
  - Sites
  - Forms
- What are the best pages?
  - Most common/most unique
  - Most commonly accessed/least commonly accessed
  - High search rank

What is Purpose of Crawl?

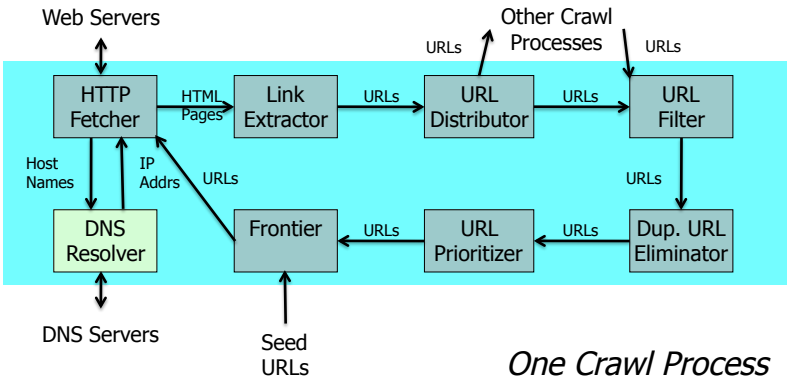
- Support a search engine
  - general
  - focused
- Web archiving
- Data mining – e.g., estimating term relatedness
- Looking for certain material
  - Use of copyrighted material
  - Mention of company or person

Challenges

- Scale: use of disk, distribution
- Crawl order
  - comprehensive
  - current
  - quality or relevance
- Good behavior
  - Don't overload sites
  - Obey exclusion rules
- Deal with bad behavior, intentional or inadvertent

Example Crawler Architecture

From: C. Olsten & M. Najork. Web Crawling. *Foundations and Trends in IR 4*(2), 2010.



One Crawl Process

The Pieces 1

- Frontier: URLs to fetch
  Priority, Politeness
- HTTP Fetcher
  - Uses DNS service to look up host names
  - Checks for Robot Exclusion Rules
  - Downloads page
- Link Extractor: Parses HTML to get URLs
  (usually doesn't execute scripts)

The Pieces 2

- URL Distributor: Assign URLs to processes based on host, domain, IP
  (many stay in same process)
- URL Filter
  - Remove blacklisted sites
  - Remove URLs with particular file extensions
- Duplicate URL Eliminator: check against URLs already discovered
- URL Prioritizer: Where does URL go in Frontier

Duplicate Detection Issues

- Can have different syntax for same page
- Can have same content at different URLs
- Examples
  - http://www.amazon.com/page.html and http://amazon.com/page.html
  - http://www.intel.com/page.html and http://www.intel.com:80/page.html
  - http://ww1.ibm.com/page.html and http://ww2.ibm.com/page.html
  - http://www.transarc.com/afs/tr/page.html and file://localhost/afs/tr/page.html

Additional Crawls Possibilities

- Archive the retrieved pages
- Eliminate near-duplicate content
  (will cover "shingling" later)
- Re-crawl links periodically

Data Structures

- For discovered URLs (URL-seen test)
  Need insert & membership test
- To be downloaded (Frontier)
  Need insert & remove next

Frontier

- Simple: FIFO queue
  - Gives breadth-first traversal
  - Problem: Long runs of URLs from same site
  - Solution: Hash URLs to multiple queues based on site or IP. Assign each queue to a (blocking) thread

Better Politeness

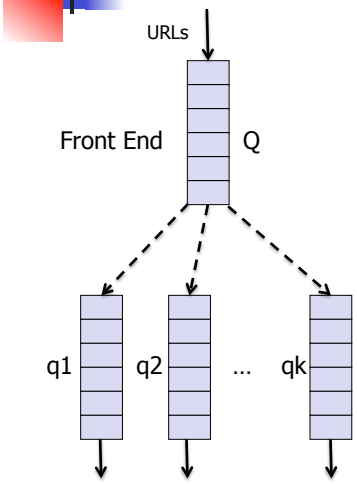
Delay requests to each server based on previous retrieval time

E.g., Page takes 0.6s to download

Wait for 5*0.6s before next request

Biases crawl towards responsive server

Mercator Frontier Structure



Hosts Table T

server	back-end queue
www.amazon.com	q1
www.pdx.edu	q17
…	…

Priority Queue

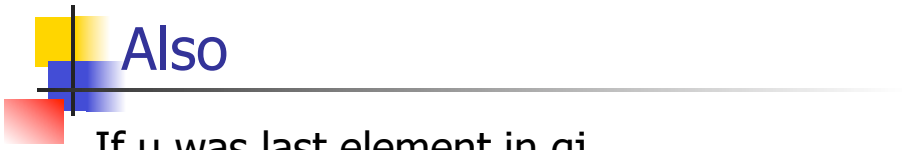
<q3,10:15>; <q6,10:16>; <q1,10:16>, …

How it Works

There are about 3k threads

Each thread does

```
Take <qj,t> from Priority Queue
Get URL u from front of queue qj
Wait until time t
Download u (taking time x)
Put <qj, now + 5*x> in Priority Queue
```

Also

If u was last element in qj

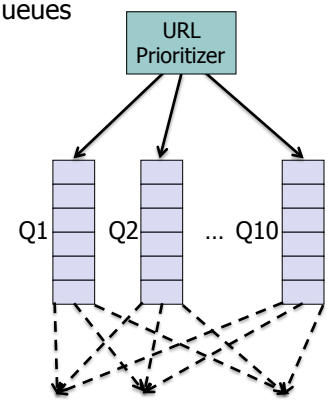
```
Delete <server(u), qj> from Hosts Table T
Repeat
  Remove URL v from Q
  Put v in qi for server(v) given by T
Until
  Find a v where server(v) not in T
Put v in qj
Add <server(v), qj> to T
```

Can Prioritize in Front End

Split Q

Q1 – highest priority; Q10 – lowest priority

Threads fetch from Q1 … Q10 using a policy biased towards higher-priority queues



URL-Seen Test (UST)

Also called Duplicate URL Elimination (DUE)

Mainly insert and membership

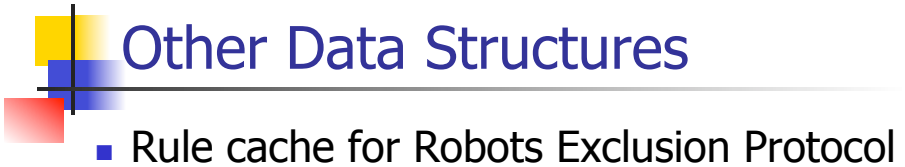
In a re-crawl setting, may need to flag bad pages

Disk-Based Hash for UST?

- Problem: Seek per look up
- Partial Solution: Cache popular URLs
- Better Solution: Do membership test & insert on a batch, make a single pass through hash table
  - Issue: Delays adding new URLs to Frontier
  - Issue: Need to remember URLs + hashes
  - Variation: Collect update batches on disk

Reducing Communication

If you also cache popular URLs mapped to other crawler processes, you can avoid some inter-process traffic



Other Data Structures

- Rule cache for Robots Exclusion Protocol
  - [See next slide]
  - Don't want to read this file before fetching each URL
- DNS cache
  - `www.pdx.edu` → `131.252.115.23`
  - DNS lookup can have high latency

Robot Exclusion File

http://www.cat.pdx.edu/robots.txt

```
User-agent: psu-gsa-crawler
Disallow: /

User-agent: *
Disallow: /administrator/
Disallow: /cache/
Disallow: /components/
Disallow: /editor/
Disallow: /help/
Disallow: /images/
Disallow: /includes/
Disallow: /language/
Disallow: /mambots/
Disallow: /media/
Disallow: /modules/
Disallow: /templates/
Disallow: /installation/
Disallow: /index.php?option=com_loudmouth
Disallow: /index.php?option=com_extcalendar
```

Crawl Ordering

- Can differ for batch crawling versus incremental crawling
  - Coverage vs. currency
- Can restart batch crawl periodically
- Factors
  - Importance of page
  - (Likely) relevance to crawl purpose
  - Dynamicity of page or site (must track changes)

Weighted Coverage (WC)

$$WC(t) = \sum_{p \in C(t)} w(p)$$

Where

- $C(t)$ = pages crawled up to time t
- $w(p)$ = weight of page p
- For example, $w(p)$ = PageRank(p)

Comprehensive Crawling

- BFS: Can be good initially with good seed set
- In-degree: How many times have I found this URL? May reflect PR
- Estimated PageRank

  Say, based on pages found so far

Other Ideas

- Prioritize sites with a lot of unfetched URLs

  Politeness limits how fast you can get through them
- Prioritize based on being ranked high in common searches
  - "Needy queries"
  - Relevance estimated based on cues in anchor text and URL