Web Crawling

Mandar Mitra

Indian Statistical Institute

Outline

What is a web crawler?

Definition

A *web crawler* / *robot* / *spider* is a system for the bulk downloading of web pages.

What is a web crawler?

Definition

A *web crawler* / *robot* / *spider* is a system for the bulk downloading of web pages.

Why might you need a crawler?

- News
- Technical documentation
- Academic papers (cf. DBLP)
- Product / movie reviews
- Social media

Basic algorithm

Input: a set of seed Uniform Resource Locators (URLs)

Steps:

1. Initialise *frontier* with seed URLs.
2. Get URL from frontier.
3. Fetch page addressed by URL.
4. Store content of downloaded page.
5. Extract hyperlinks contained in the downloaded page.
6. Add extracted links to frontier.

Basic algorithm

Input: a set of seed Uniform Resource Locators (URLs)

Steps:

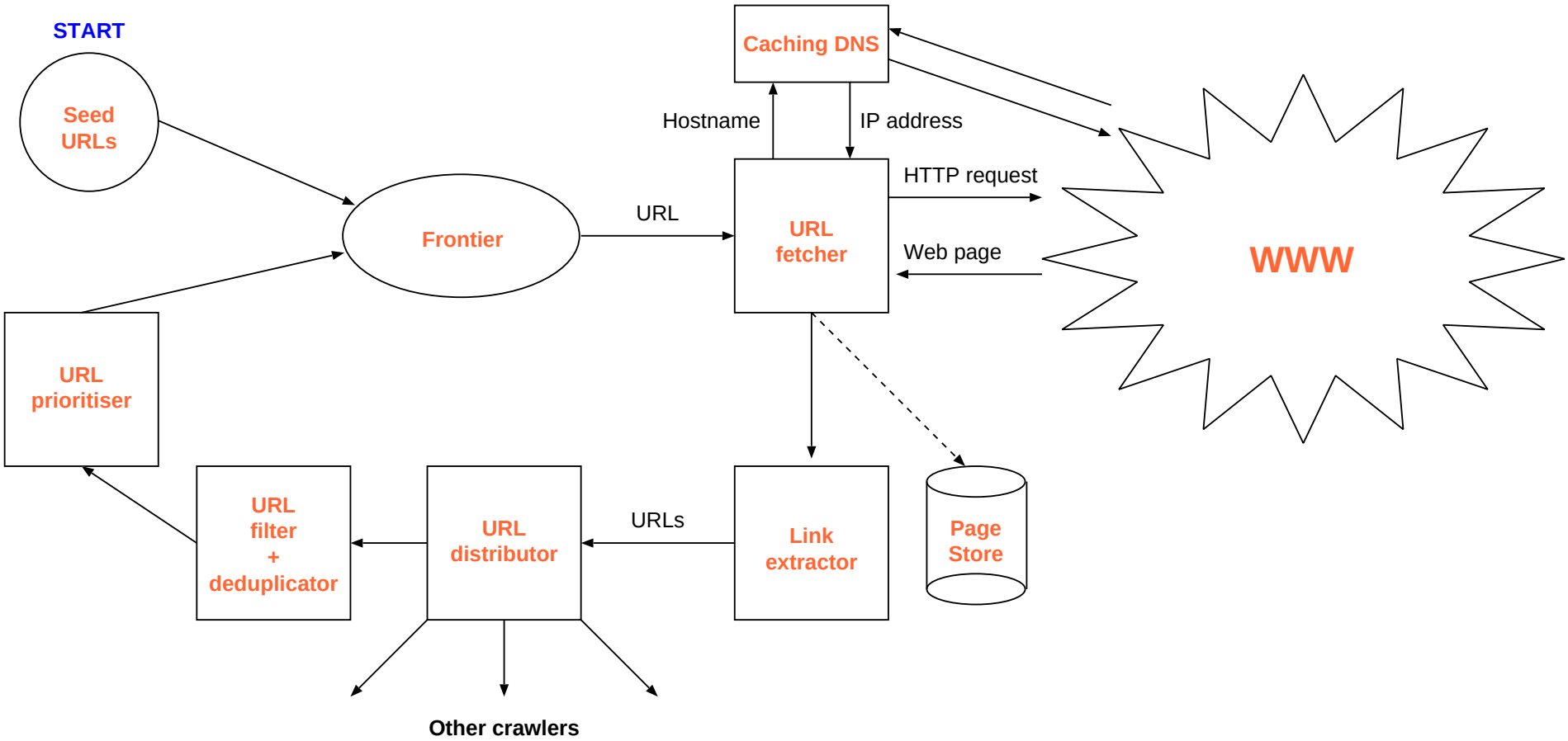
1. Initialise *frontier* with seed URLs.
2. Get URL from frontier.
3. Fetch page addressed by URL.
4. Store content of downloaded page.
5. Extract hyperlinks contained in the downloaded page.
6. Add extracted links to frontier.
7. Repeat steps 2 to 6 until done.

Challenges

- Content selection
  - content type: news, movie reviews, . . .
  - content quality: high-quality vs. low-quality, malicious / adversarial content
  - deduplication
- Scheduling tradeoffs: coverage vs. freshness; exploration vs. exploitation
  - high-value content should be obtained early
  - dynamic content should be updated quickly
- Scalability: computing + storage + bandwidth
- Politeness
  - per-site rate limitations to avoid denial-of-service like attacks
  - robot exclusion policies

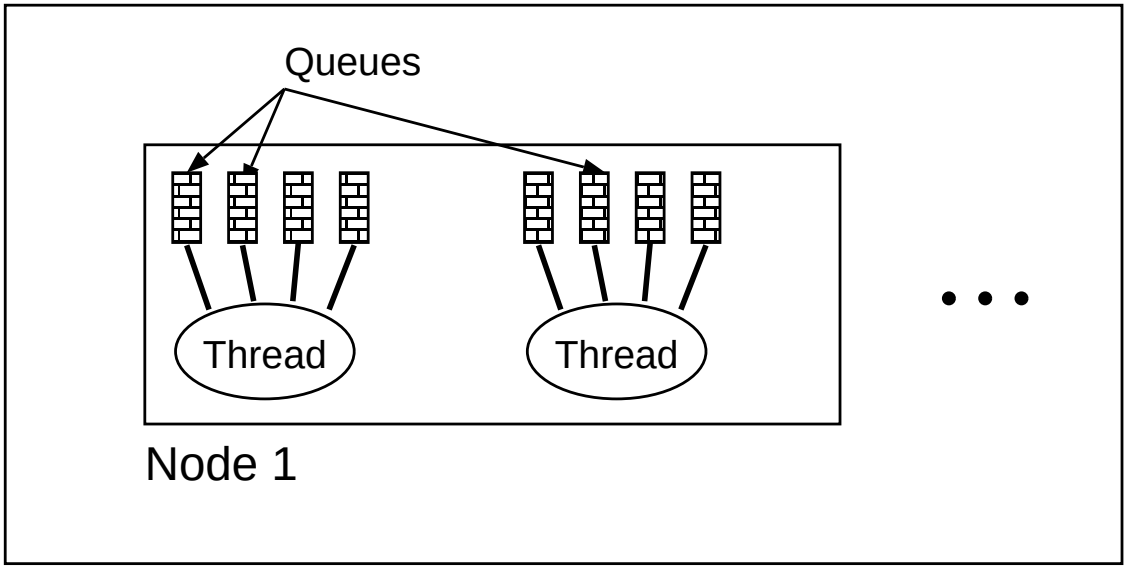
Outline

Overall architecture



Distributed crawler components

Underlying hardware

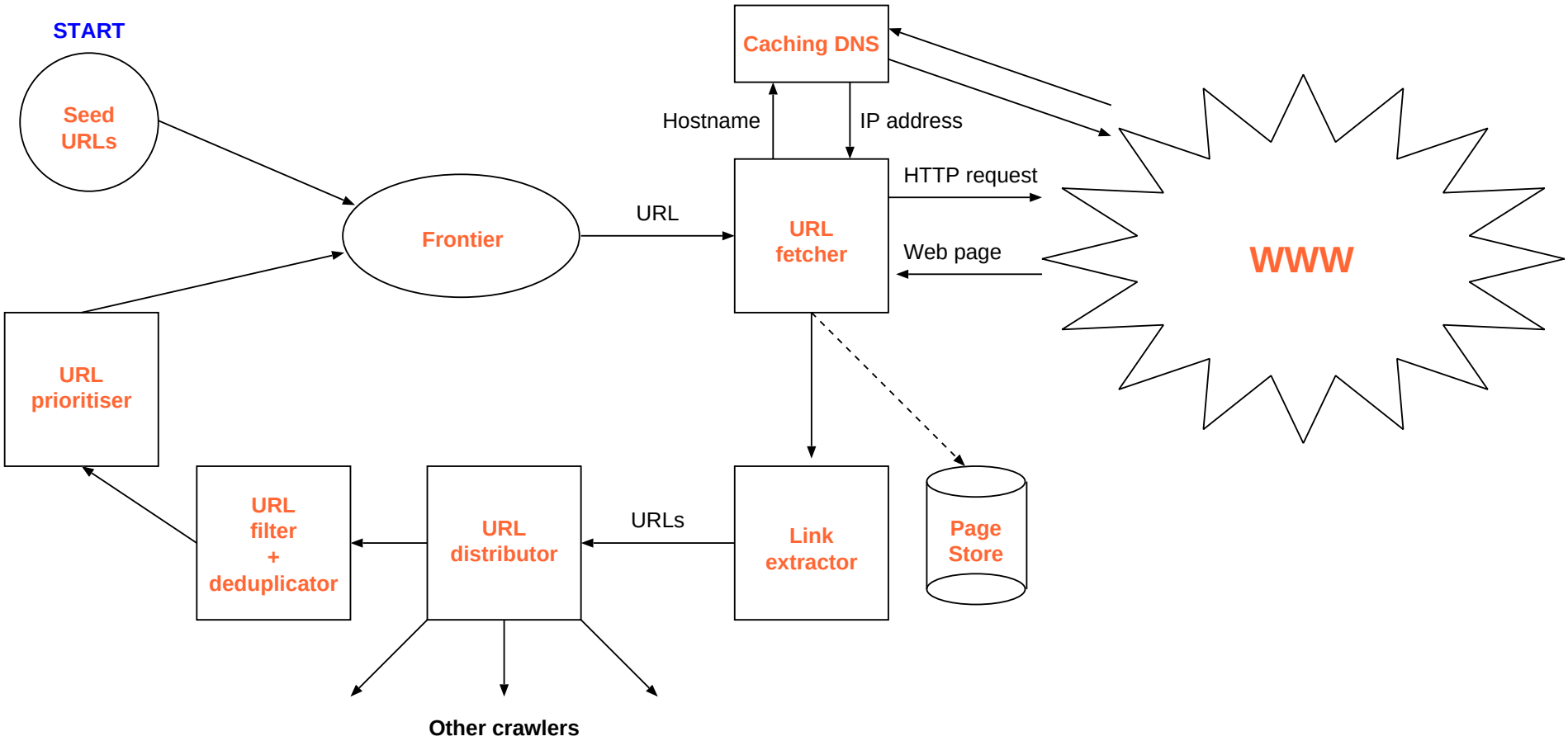


- Multiple machines / nodes inter-connected by high-speed network
- Multiple crawler threads / processes per machine
- Multiple fetch queues per thread
- Each node responsible for a subset of all URLs to be crawled

Distributed crawler components

- Distributor: assign URLs to a particular node
- URL space partitioned across "web site" boundaries (symbolic host name, domain name, same IP address)
- URL host / domain / IP address $\xrightarrow{hash}$ node / crawler queue
- Data structures also partitioned
- Tries to reduce inter-node traffic
- Single crawler thread handles multiple queues
- Keeps crawler busy without sending overlapping / successive requests to the same server
- Possible politeness policy: next request to server X delayed by 10 times the time taken to download most recent page from server X.
  $\Rightarrow$ more load on powerful servers

Overall architecture



Distributed crawler components

- Filtering: exclude
  - URLs from black-listed sites
  - URLs with file extensions that are not required
- Deduplication: ignore URLs processed already
  - also called the *URL-seen test (UST)* or *duplicate URL eliminator (DUE)*
  - more on this later
- Prioritisation / crawl ordering
  - breadth-first search
  - page importance (indegree, PageRank, . . .)
  - rate of change

Auxiliary data structures

Robots Exclusion Protocol

- `/robots.txt` used by administrators to (selectively) prohibit crawling
  - example

```
# Group 1
User-agent: Googlebot
Disallow: /nogooglebot/

# Group 2
User-agent: *
Allow: /

Sitemap: http://www.example.com/sitemap.xml
```

- Crawlers cache `robots.txt` (as for DNS entries)

Outline

Scoped / topical / focused crawling

- Only pages belonging to particular *category* (**relevant** pages) are of interest
  - sports, entertainment, etc.
  - geography / region / language
  - format (images, audio files, etc.)
  - genre (scholarly literature, course material, etc.)
  - . . .

Scoped / topical / focused crawling

- Only pages belonging to particular *category* (**relevant** pages) are of interest
  - sports, entertainment, etc.
  - geography / region / language
  - format (images, audio files, etc.)
  - genre (scholarly literature, course material, etc.)
  - . . .

Principle

Relevant pages tend to link to other relevant pages, either directly or via short chains of links.

Focused crawling - I

Fish search

- Each crawled page p classified as relevant / irrelevant
- Neighbourhood of relevant pages explored up to some depth d

Focused crawling - II

Improved fish search

- Neighbourhoods of relevant pages explored in non-uniform fashion (most promising links explored first)
- Relevance score of uncrawled neighbour estimated using
  - anchor text + nearby text
  - portion of crawled neighbour(s)

Focused crawling - III

Using taxonomies

- Use pre-existing topic taxonomies (e.g., Open Directory Project / `dmoz` (now closed))
  - sample web pages organised into hierarchical categories
- Train classifier for taxonomy
- Scope defined as set of taxonomy nodes
- Crawler preferentially follows “relevant” pages + pages from parent categories

Focused crawling: empirical observations

- Seeding with topical pages generally ineffective
- Different focused crawlers, started from different seed sets, "converge" to a significant extent.

Problematic content - I

Crawler traps

- Large (possibly infinite) URL space auto-generated by web site
  - example: calendar type pages with *previous* / *next* links

Problematic content - I

Crawler traps

- Large (possibly infinite) URL space auto-generated by web site
  - example: calendar type pages with *previous* / *next* links
- Remedy: *Budget Enforcement with Anti-Spam Tactics* (BEAST)
  - assigns a budget to each web site
  - prioritizes URLs from site based on remaining budget and/or reputation

Problematic content - II

Web spam

- *Keyword stuffing:* adding highly searched terms
- *Link spam:* creating cliques to mislead link-based ranking algorithms like PageRank
- *Cloaking:* serving very different content to crawlers and humans
  - Most web crawlers do not execute JavaScript
  - Include sandboxed + lightweight JavaScript parsers and execution engines in crawler?

Reference

Web Crawling

Christopher Olston and Marc Najork

Foundations and Trends in Information Retrieval, Vol. 4, No. 3, pp. 175–246

2010