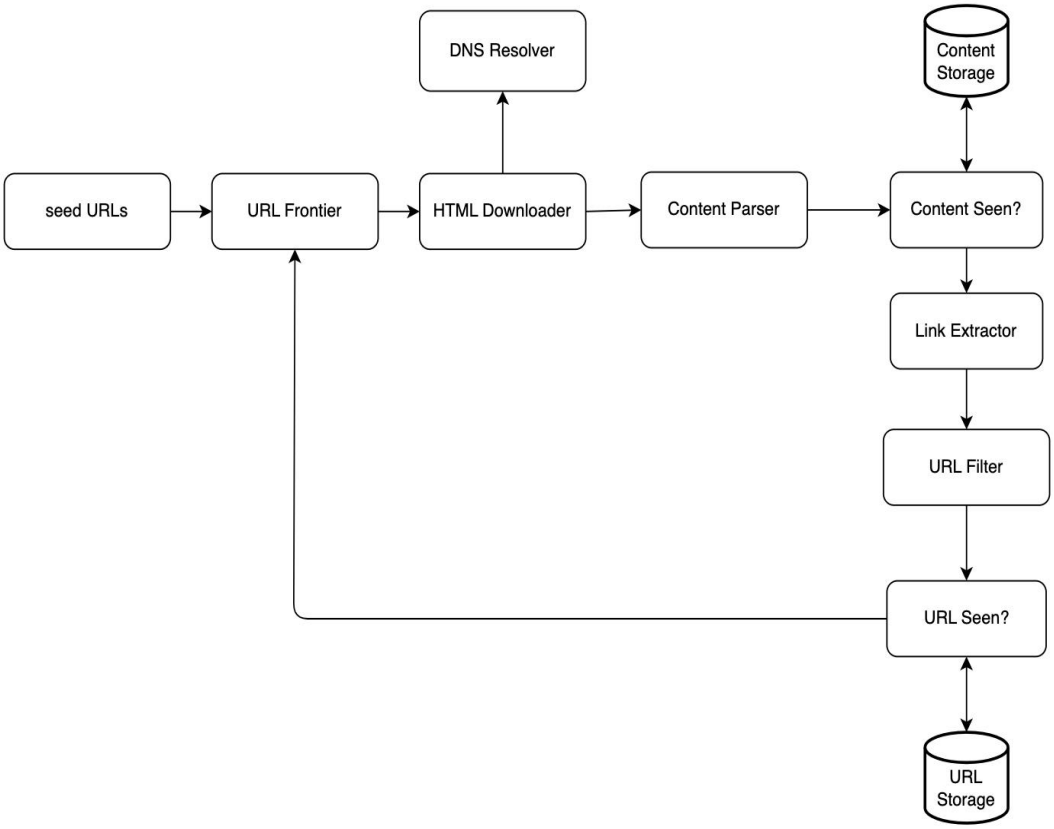
46. Web Crawler

Requirements clarification

- **Functional requirements**
  - Given a set of URLs, download all the web pages addressed by the URLs.
  - Track newly added or edited web pages.
  - Prioritize web pages that are dynamic as these pages appear more frequenctly in search engine ranking.
- **Non-functional requirements**
- **High scalability**
  - Web crawling should be extremely efficient using parallelization.
- **Robustness**
  - The crawler must handle all edge cases, like bad HTML, unresponsive servers, crashes, malicious links, etc.
- **Politeness**
  - The crawler should not make too many requests to a website within a short time interval.
- **Extensibility**
  - The system is flexible so that minimal changes are needed to support new content types (images, videos).

High-level design



Seed URLs

- A web crawler uses seed URLs as a starting point for the crawl process.
- The general strategy is to divide the entire URL space into smaller ones.
  - Based on locality
  - Based on topics

URL Frontier

- Stores URLs need to be downloaded.
- A First-in-First-out (FIFO) queue.

HTML Downloader

- Downloads web pages from the internet.

DNS Resolver

- The HTML Downloader calls the DNS Resolver to get the corresponding IP address for the URL.

Content Parser

- Parses and validate the content of web pages.

Content Seen?

- Detects new content previously stored in the system.
- Eliminates data redundancy and shorten processing time.
- To compare two two web pages, it compares the hash values of them.

Content Storage

- Stores HTML content.
- Most of the content is stored on disk.
- Popular content is kept in memory to reduce latency.

Link extractor

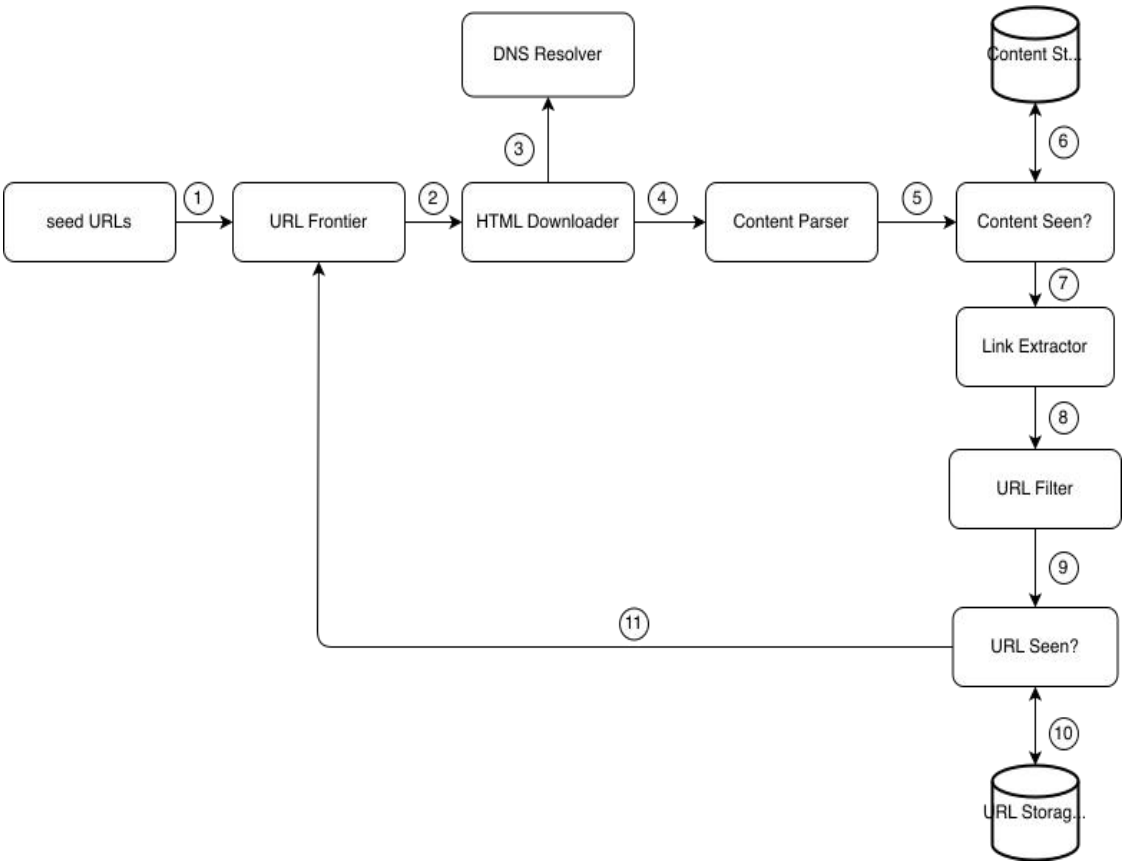
- Parses and extracts links from HTML pages.

URL Filter

- Excludes certain content types, file extensions, error links and URLs in “blacklisted” sites.
- URL Storage
- Stores already visited URLs.

Detailed design

Workflow



- Step 1: Add seed URLs to the URL Frontier
- Step 2: HTML Downloader fetches a list of URLs from URL Frontier.
- Step 3: HTML Downloader gets IP addresses of URLs from DNS resolver and starts downloading.
- Step 4: Content Parser parses HTML pages and checks if pages are malformed.
- Step 5: After content is parsed and validated, it is passed to the “Content Seen?” component.
- Step 6: “Content Seen” component checks if a HTML page is already in the storage.

- If it is in the storage, this means the same content in a different URL has already been processed. In this case, the HTML page is discarded.
- If it is not in the storage, the system has not processed the same content before. The content is passed to Link Extractor.

- Step 7: Link extractor extracts links from HTML pages.
- Step 8: Extracted links are passed to the URL filter.
- Step 9: After links are filtered, they are passed to the “URL Seen?” component.
- Step 10: “URL Seen” component checks if a URL is already in the storage, if yes, it is processed before, and nothing needs to be done.
- Step 11: If a URL has not been processed before, it is added to the URL Frontier.

Algorithm

Choice

BFS (DFS is usually not a good choice because the depth of DFS can be very deep).

URL frontier

Politeness

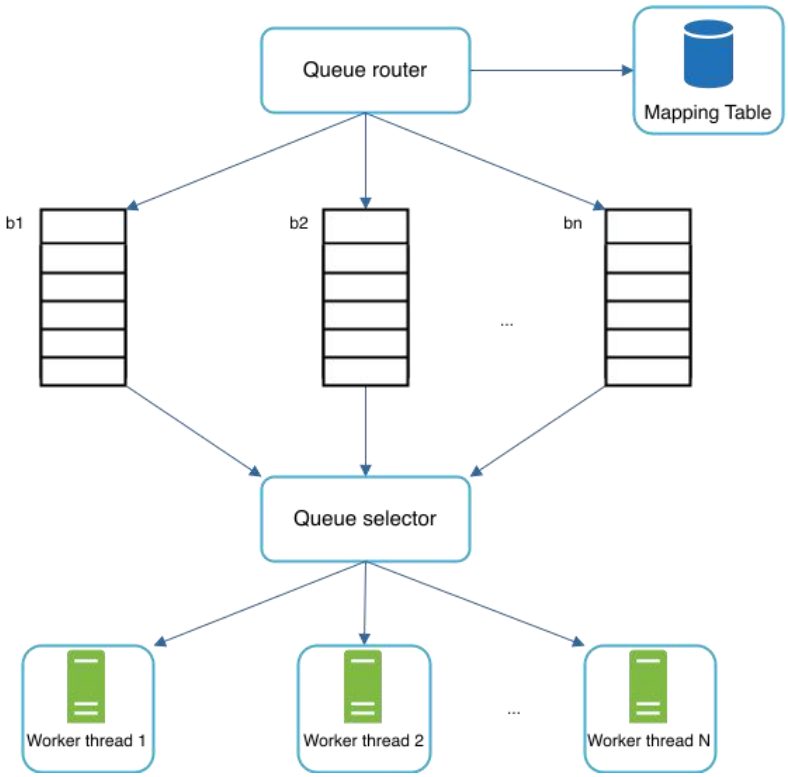
Concept

- A web crawler should avoid sending too many requests to the same hosting server within a short period.

Solution

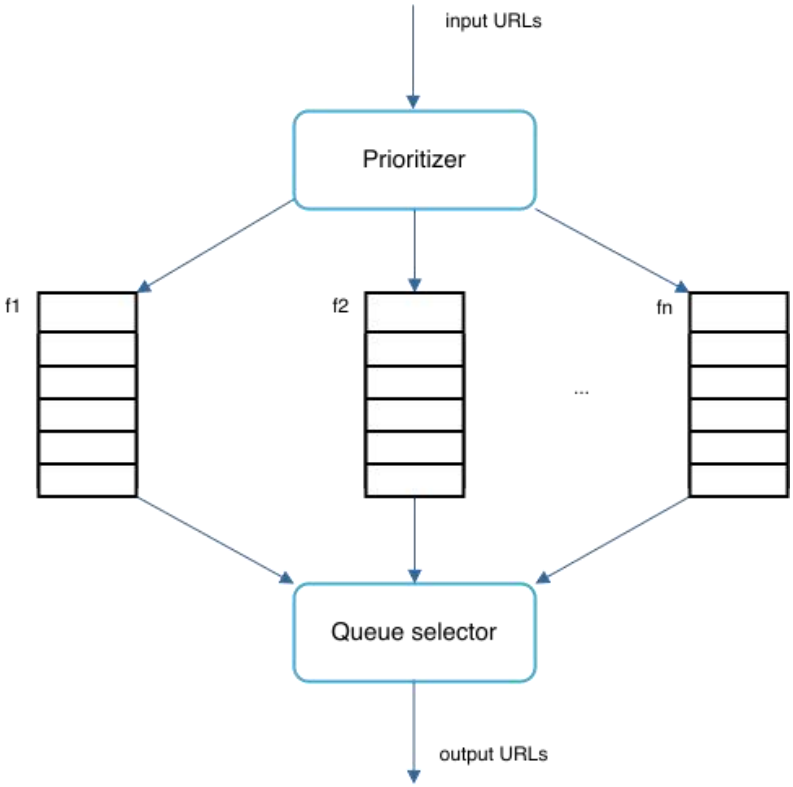
- Download one page at a time from the same host. A delay can be added between two download tasks.

- **Implementation**
  - Queue router: It ensures that each queue (b1, b2, … bn) only contains URLs from the same host.
  - Mapping table: It maps each host to a queue.
  - FIFO queues (b1, b2, ..., bn): Each queue contains URLs from the same host.
  - Queue selector: Each worker thread is mapped to a FIFO queue, and it only downloads URLs from that queue. The queue selection logic is done by the Queue selector.
  - Worker thread (1,2,..., N): A worker thread downloads web pages one by one from the same host. A delay can be added between two download tasks.



Priority

- **Factors**
  - PageRank
  - Website traffic
  - Update frequency
- **Implementation**
  - Prioritizer: It takes URLs as input and computes the priorities.
  - Queue (f1 to fn): Each queue has an assigned priority. Queues with high priority are selected with higher probability.
  - Queue selector: Randomly choose a queue with a bias towards queues with higher priority.



Freshness

Concepts

- Web pages are constantly being added, deleted, and edited. A web crawler must periodically recrawl downloaded pages to keep our data set fresh.

Strategies

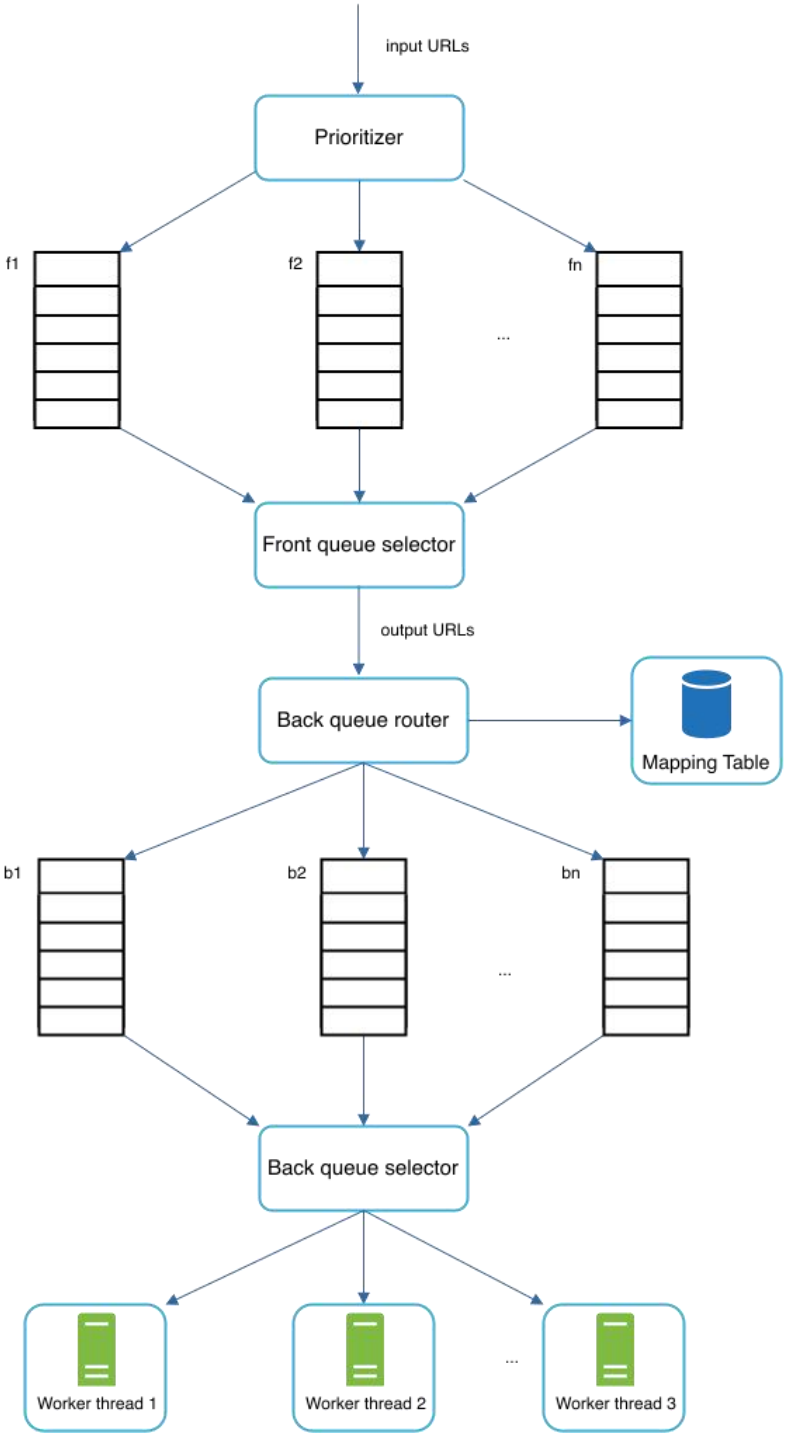
- Recrawl based on web pages' update history.
- Prioritize URLs and recrawl important pages first and more frequently.

Storage

- **Strategies**
  - Hybird approach
- **Disk:**
  - The majority of URLs
- **Memory:**
  - Maintains buffersfor enqueue/dequeue operations.
  - Data in the buffer is periodically written to the disk.

Final structure

- Front queues: Manage prioritization
- Back queues: Manage politeness



HTML Downloader

Robots exclusion protocol

- The file called robot.txt, a standard used by websites to communicate with crawlers.
- It specifies what pages crawlers are allowed to download.
- Before attempting to crawl a web site, a crawler should check its corresponding robots.txt first and follow its rules.
- To avoid repeat downloads of robots.txt file, we cache the results of the file. The file is downloaded and saved to cache periodically.

Performance optimization

- Crawl jobs are distributed into multiple servers (downloader), and each server runs multiple threads.
- Maintains our DNS cache to avoid calling DNS (bottleneck) frequently. Our DNS cache is updated periodically by cron jobs.
- Deploys crawl servers geographically closer to website hosts.
- Uses short timeout when crawling web pages.

Robustness optimization

- Crawl servers should save crawl states and data so that A disrupted crawl can be restarted easily.
- Crawl servers must handle exceptions gracefully without crashing the system.

Detect and avoid problematic content

- Use hashes or checksums help to detect duplication
- Setting a maximal length for URLs avoids spider traps (a web page that causes a crawler in an infinite loop).
- Excludes advertisements, code snippets, spam URLs, etc.

Key points

- Politeness: Download one page at a time from the same host. A delay can be added between two download tasks.
- Priority: Use multiple queues to store URLs in different priorities, randomly choose a queue with a bias towards queues with higher priority.
- Freshness: A web crawler must periodically recrawl downloaded pages based on web pages' update history and importance.
- Need to consider performance, robustness, problematic web pages for crawlers.