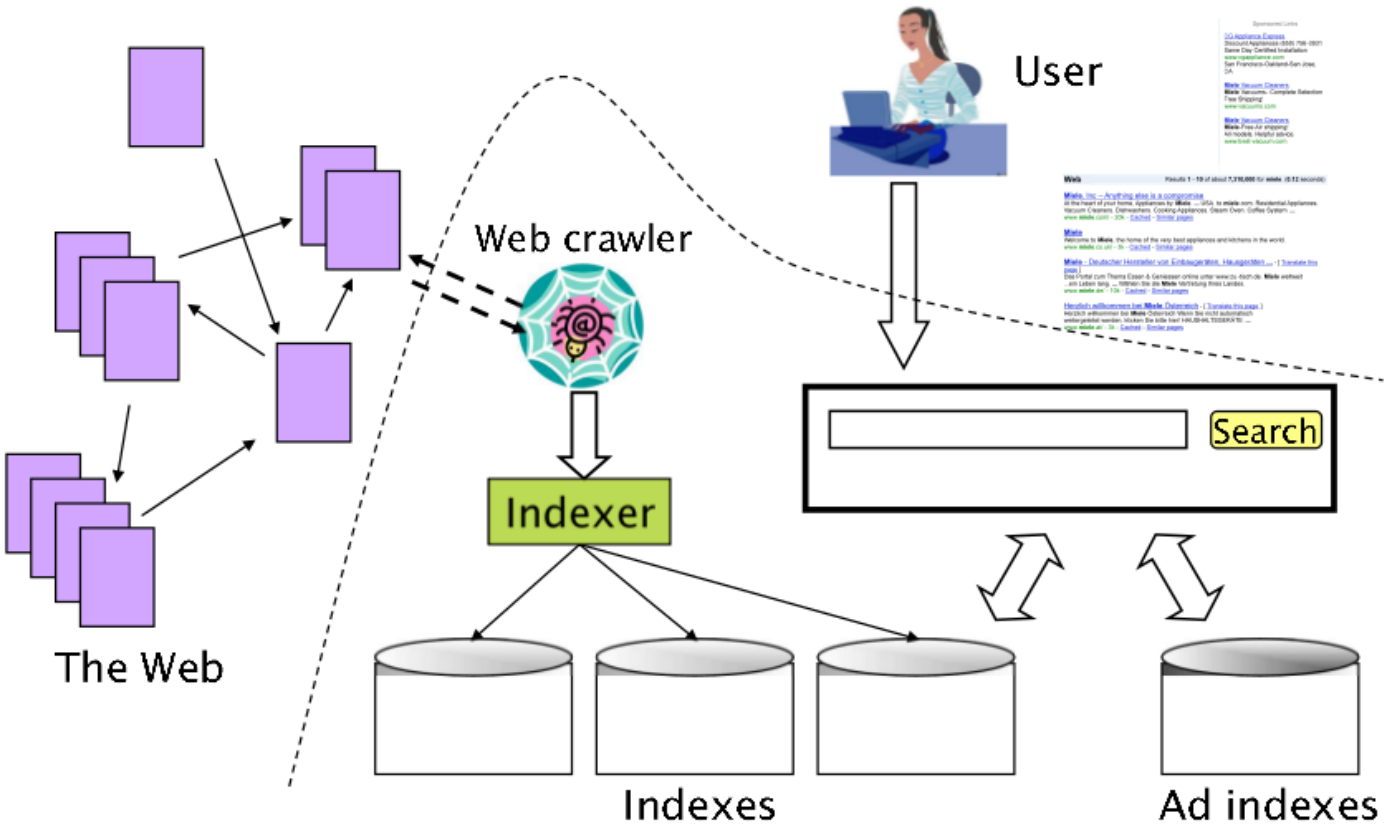
Поисковый робот

Лекция 5

БГУ ФПМИ, 2018

План


User
Web crawler
Search
Indexer
The Web
Indexes
Ad indexes

Поисковый робот (Web Crawler)

Чего хотим?

Находить и скачивать документы из Интернета в автоматическом режиме.

Поисковый робот (Web Crawler)

Чего хотим?

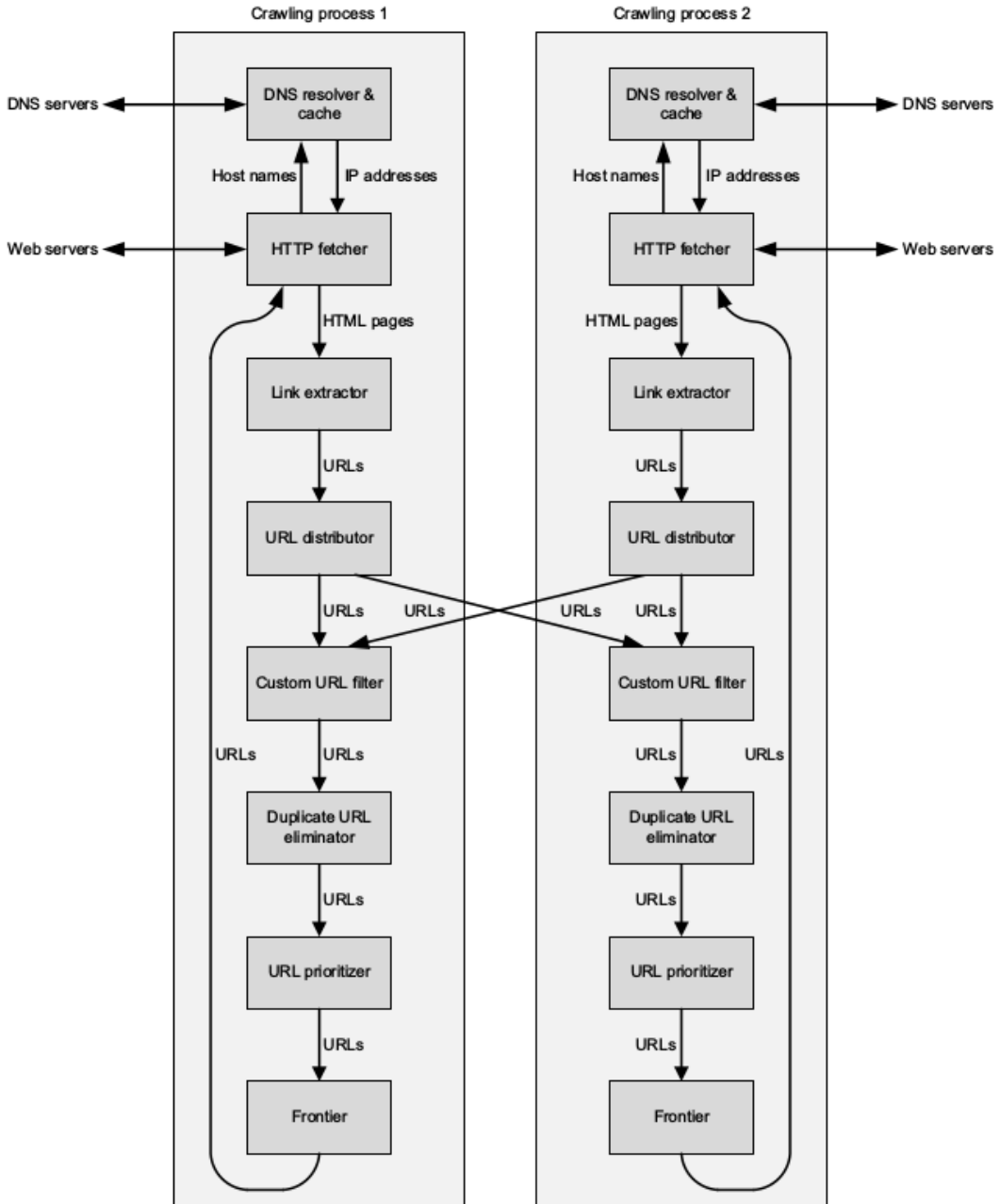
Находить и скачивать документы из Интернета в автоматическом режиме.

- Интернет огромен и постоянно растет.
- Интернет не контролируется поисковыми системами.
- Документы постоянно меняются.

План

Требования

- Учтойчивость (к spider traps).
- Вежливость.
- Распределенность.
- Масштабируемость.
- Эффективность.
- Качество (покрытие потенциальных информационных потребностей).
- Свежесть.
- Расширяемость (модульность).


Crawling process 1
Crawling process 2
DNS servers
DNS resolver & cache
DNS servers
Host names
IP addresses
Host names
IP addresses
Web servers
HTTP fetcher
Web servers
HTML pages
HTML pages
Link extractor
Link extractor
URLs
URLs
URL distributor
URL distributor
URLs
URLs
URLs
URLs
Custom URL filter
Custom URL filter
URLs
URLs
URLs
URLs
Duplicate URL eliminator
Duplicate URL eliminator
URLs
URLs
URL prioritizer
URL prioritizer
URLs
URLs
Frontier
Frontier

План

URL

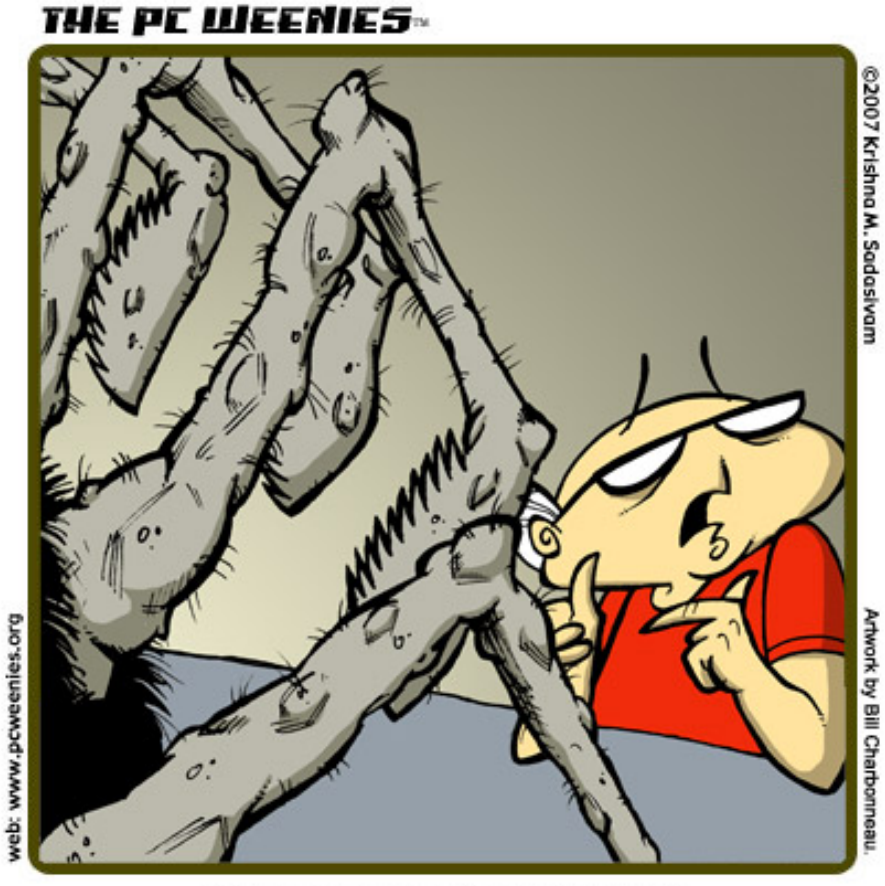
- Каждый документ имеет уникальный Uniform Resource Locator.
- Документы хранятся на серверах, которые используют протокол HTTP(S) для взаимодействия с клиентами.

<схема>://<логин>:<пароль>@<хост>:<порт>/<путь>
?<параметры>#<якорь>

- https://yandex.by/search/?text=web%20crawl&lr=157
- https://en.wikipedia.org/wiki/Eric_Clapton#Guitars
- https://172.217.20.174

Скачивание отдельного документа

1. Робот обращается к DNS, чтобы узнать IP-адрес хоста.
2. Устанавливает соединение с хостом (используя заданный порт).
3. Отправляет HTTP запрос для получения документа (как правило GET запрос).



"Can somebody call our SEO guy?
I don't think Google is crawling our site properly."

Robots.txt

- Робот может создавать значительную нагрузку на сайт.
- Не все разделы позволено обходить.

Стандарт исключений для роботов (robots.txt) — протокол и соотвествтующий файл в корне сайта, определяющий взаимодействие с поисковыми роботами.

Robots.txt

Состоит из стандартных директив User-agent, Disallow и нестандартных (могут поддерживаться не всеми роботами): Allow, Host, Sitemap, Crawl-delay и др.

Пример: запрет доступа всех роботов ко всему сайту

User-agent: *
Disallow: /

https://ru.wikipedia.org/robots.txt

```
...
User-agent: Download Ninja
Disallow: /

# Misbehaving: requests much too fast:
User-agent: fast
Disallow: /

#
# Sorry, wget in its recursive mode is a frequent problem.
# Please read the man page and use it properly; there is a
# --wait option you can use to set the delay between hits,
# for instance.
#
User-agent: wget
Disallow: /
...

# Страницы и подстраницы участников и участниц
Disallow: /wiki/%D0%A3%D1%87%D0%B0%D1%81%D1%82%D0%BD%D0%B8%D0%BA:*
Disallow: /wiki/%D0%A3%D1%87%D0%B0%D1%81%D1%82%D0%BD%D0%B8%D1%86%D0%B0:*

# Страницы и подстраницы обсуждения участников и участниц
Disallow: /wiki/%D0%9E%D0%B1%D1%81%D1%83%D0%B6%D0%B4%D0%B5%D0%BD%D0%B8%D0
Disallow: /wiki/%D0%9E%D0%B1%D1%81%D1%83%D0%B6%D0%B4%D0%B5%D0%BD%D0%B8%D0

# ВП:Выборы арбитров/
Disallow: /wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D
```

Robots.txt: fun

https://www.google.ru/humans.txt

https://www.google.ru/killer-robots.txt

User-Agent: T-1000
User-Agent: T-80
Disallow: /+LarryPage
Disallow: /+SergeyBrin

Sitemap

Sitemap — XML-файл с информацией об индексируемых страницах сайта.

- Содержит информацию о страницах, которые робот не сможет обнаружить сам.
- Содержит вспомогательную информацию о времени последнего обновления и частоте обновления документов.

http://www.vpoxod.ru/sitemap.xml

urlset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http:/
<!-- www.check-domains.com sitemap generator -->
<url>
<loc>http://www.vpoxod.ru/</loc>
<lastmod>2015-03-23T10:03:25+00:00</lastmod>
<changefreq>monthly</changefreq>
<priority>1.0000</priority>
</url>
...
<url>
<loc>
http://www.vpoxod.ru/route/elbrus/elbrus-s-severa/about
</loc>
<lastmod>2015-03-23T10:03:25+00:00</lastmod>
<changefreq>monthly</changefreq>
<priority>0.5120</priority>
</url>

Поставщики документов

- Многие документы публикуются: создаются в заданный момент времени и редко обновляются (новостные статьи, почтовые сообщения).
- Опубликованные документы могут быть организованы в виде порций (feed):
  - push feed: оповещение подписчика о новых публикациях.
  - pull feed: подписчик сам периодически проверяет публикации (RSS).

https://lenta.ru/rss/top7

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
  <channel>
    <language>ru</language>
    <title>Lenta.ru</title>
   <description>Новости, статьи, фотографии, видео. Семь дней в неделю, 24 часа в сутки</descriptio
    <link>http://lenta.ru</link>
    <image>
      <url>http://assets.lenta.ru/small_logo.png</url>
      <title>Lenta.ru</title>
      <link>http://lenta.ru</link>
      <width>134</width>
      <height>22</height>
    </image>
    <atom:link rel="self" type="application/rss+xml" href="http://lenta.ru/rss/top7"/>
<item>
  <guid>https://lenta.ru/news/2017/03/09/barcawins/</guid>
 <title>«Барселона» забила ПСЖ шесть мячей и вышла в четвертьфинал Лиги чемпионов</title>
  <link>https://lenta.ru/news/2017/03/09/barcawins/</link>
  <description>
  ...
```

Сегментация страниц

- Многие страницы в Интернете содержат текст, ссылки и т.д., которые не относятся к основному содержимому страницы.
- Это негативно сказывается на качестве поиска.
- Важно уметь выделять на странице блоки с содержанием, отражающим тему документа.

DOM

Document Object Model — не зависящий от платформы и языка программный интерфейс, позволяющий получить доступ к содержимому HTML и XML документов.

- Документ представляется в виде дерева.
- Нужные атрибуты могут быть найдены путем прохода по дереву.

Эмпирическое выделение основного текста документа

Предположение

Основной текст документа располагается в той части HTML, которая содержит много токенов, не являющихся HTML-тегами. При этом остальной текст HTML-страницы содержит большинство её тегов.

Эмпирическое выделение основного текста документа

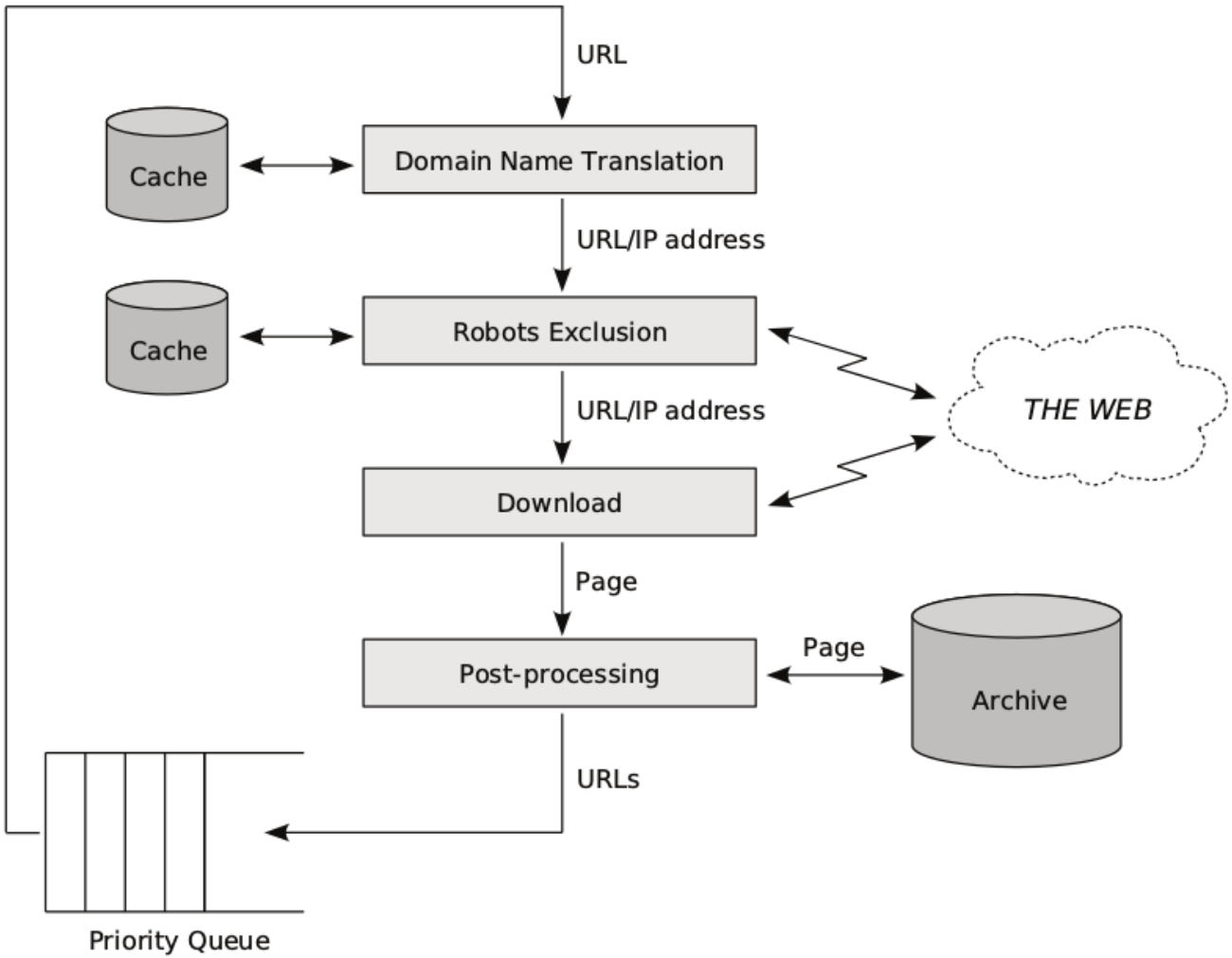
Представим документ в виде последовательности бит: $b_i = 1$, если i-ый токен — HTML-тег.

$$\sum_{n=0}^{i-1} b_n + \sum_{n=i}^{j}(1 - b_n) + \sum_{n=j+1}^{N-1} b_n \to \max_{i,j}$$

(i, j) — границы выделенного текста.

Процесс скачивания Интернета

1. Сформировать очередь из URL-ов источников (seeds).
2. Взять URL из очереди, скачать документ.
3. Ссылки из скаченного документа добавить в очередь, перейти к п. 2.
4. Прекратить процесс, если очередь пуста, или закончилось место на диске.


URL
Cache
Domain Name Translation
URL/IP address
Cache
Robots Exclusion
THE WEB
URL/IP address
Download
Page
Post-processing
Page
Archive
URLs
Priority Queue

План

Требования

- Прямой доступ к содержимому по URL.
- Эффективный доступ к тексту, например для построения сниппетов.
- Сжатие больших файлов.
- Эффективное обновление:
  - обработка большого количества изменений
  - добавление новых ссылок.

Эвристика: большие файлы

Лучше хранить много документов в одном большом файле, чем хранить каждый документ отдельно в маленьком файле:

- меньше накладные расходы на открытие/закрытие
- снижение времени поиска по отношению к времени чтения.

Bigtable

Хранилище робота, разработанное в Google в 2005 году.
Open-source аналоги:

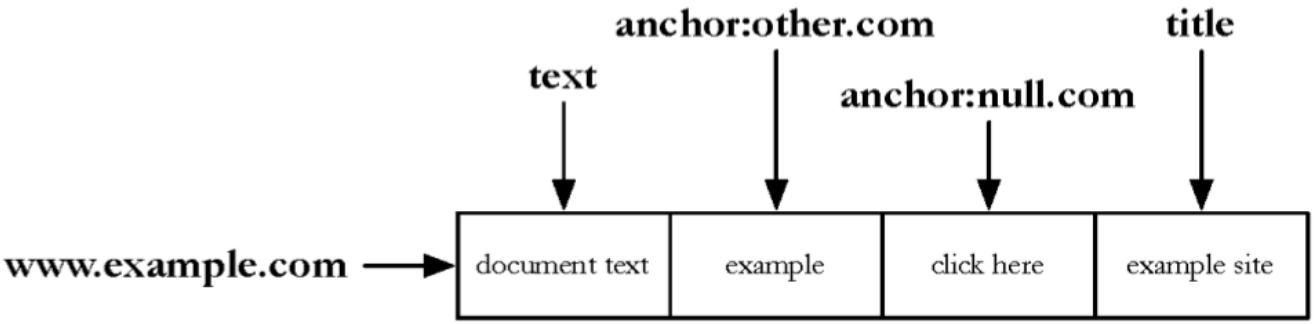
- Apache Cassandra
- Apache HBase
- Hypertable

Bigtable

- Логически представлена в виде строк.
- Хранится распределенно в виде набора таблеток (tablets).
- Каждая ячейка задается набором из ключа строки, ключа столбца и timestamp.

Bigtable: web crawler

Строка хранит данные для одного документа



Bigtable

- Транзакции на уровне строк.
- Все изменения логируются.
- Репликация данных.
- Отказоустойчивость.
- Масштабируемость.

План

Свежесть

- Документы постоянно добавляются, удаляются, изменяются.
- В поисковой системе должна быть всегда актуальная копия документа.
- Для этого поисковый робот постоянно переобходит уже скаченные документы и сохраняет изменения в хранилище.

Свежесть

HTTP-запрос HEAD позволяет проверить, когда был изменен документ:

```
~$ curl -IHEAD 'https://en.wikipedia.org/wiki/Bigtable'
HTTP/1.1 200 OK
Date: Wed, 07 Mar 2017 23:58:39 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Server: mw1172.eqiad.wmnet
.....
```

Невозможно таким образом проверять все документы.

Возраст

Вместо бинарной характеристики свежести будем рассматривать ожидаемый возраст страницы через t дней после последнего обхода:

$$Age(t) = \int_0^t P(\text{page changed at time x})(t - x)dx$$

Возраст

Вместо бинарной характеристики свежести будем рассматривать ожидаемый возраст страницы через t дней после последнего обхода:

$$Age(t) = \int_0^t P(\text{page changed at time x})(t - x)dx$$

Предполагают, что страницы обновляются согласно экспоненциальному закону:

$$Age(\lambda, t) = \int_0^t \lambda e^{-\lambda x}(t - x)dx$$

Очередь обхода

Простейшая реализация: FIFO.

- В результате – обход в ширину, что может сказаться на нагрузке на сайт.
- Не все документы одинаково важны для переобхода.

Очередь обхода

Простейшая реализация: FIFO.

- В результате – обход в ширину, что может сказаться на нагрузке на сайт.
- Не все документы одинаково важны для переобхода.

Priority queue?

- Время операций – $O(\log n)$, что при обращениях к диску сильно влияет на скорость.

Очередь обхода

Решение

Дискретизируем значения приоритета (например, до 10-100 значений), для каждого дискретного уровня приоритета заведем свою FIFO-очередь.

Приоритизация

- Статические характеристики веба (PageRank, размер сайта, ссылки).
- Динамические характеристики (возраст, частота обновлений).
- Пользовательские данные.

План

Мотивация

30% документов являются дубликатами или очень похожими на остальные 70%.

- Меньше нагрузка на сеть и сервера.
- Уменьшение размера индекса.
- Незадублированность выдачи.
- Лучше ранжирование за счет переноса сигнала с неглавных страниц.

Виды дубликатов

Точные дубли

Бинарное совпадение документов. Обнаруживается подсчетом и сравнением сигнатур при скачивании (MD5, CRC, SHA-2).

Нечеткие дубликаты (полудубли)

Документы, отличающиеся друг от друга незначительно в некотором смысле.

Дубли, определенные вебмастерами

- Редиректы. Автоматическая переадресация на другой URL. Различают постоянную (301), временную (302/303/307), по метатегу http-equiv=”refresh”, посредством JavaScript.
- rel=”canonical”. Явное указание канонического документа в заголовке посредством атрибута.

  <link rel="canonical" href="https://blog.example.com/" />

Зеркала (mirror hosts)

- Физическая копия данных одного сервера (высокий процент совпадения) на другом.
- Доменные зеркала — соответствие нескольких доменных имен одному серверу.
- Зачастую необходимо обнаруживать автоматически, аналогично нечетким дубликатам.

Нечеткие дубликаты: идея алгоритмов

- Генерируются гипотезы, выраженные в виде вычисляемых по документам хэшей.
- Совпадение хэшей означает высокую вероятность дублирования контента.

I-Match

- По всей коллекции документов строится словарь L.
- Из словаря удаляются слова с очень высоким и очень низким idf.
- Для каждого документа ищется пересечение его уникальных термов с термами из L.
- Если размер пересечения больше порога, то результат сортируется и для него вычисляется I-Match сигнатура (SHA-1).
- Документы похожи, если их I-Match сигнатуры совпадают.

I-Match (модификация)

- Строим много словарей (100-200), из каждого случайно удаляем 30-40% слов.
- По каждому словарю вычисляем I-Match сигнатуру.
- Если для пары документов хотя бы 1 сигнатура совпадает, то документы — дубликаты.

TF-IDF based

- Вычисляем TF-IDF для каждого терма документа.
- Выписываем упорядоченно топ-K с наибольшим весом.
- Вычисляем сигнатуру от полученной строки.

Шинглы (shingles)

Шингл — последовательность термов документов длины k.
Обычно в задаче поиска дубликатов $k = 4$.

a rose is a rose is a rose:

- a rose is a
- rose is a rose
- is a rose is

Метод шинглов

- Документ — множество шинглов $S(d)$.
- Похожесть d_1 и d_2 выражается мерой Жаккарда между $S(d_1), S(d_2)$.
- Если мера больше заданного порога (0.9), то документы – дубли.
- НО вычисление попарной меры для всех документов нереально.

Метод шинглов (Min-Hash)

1. Вычислим для каждого шингла 64-битный хэш.
2. Вычисленные хэши отобразим с помощью случайной перестановки π, заданной на всех 64-битных числах.
3. В полученном множестве чисел выберем минимальное: x_d^π.

Теорема

$$J(S(d_1), S(d_2)) = P(x_1^\pi = x_2^\pi) \tag{1}$$

Для оценки вероятности в (1) рассмотрим 100 разных перестановок.

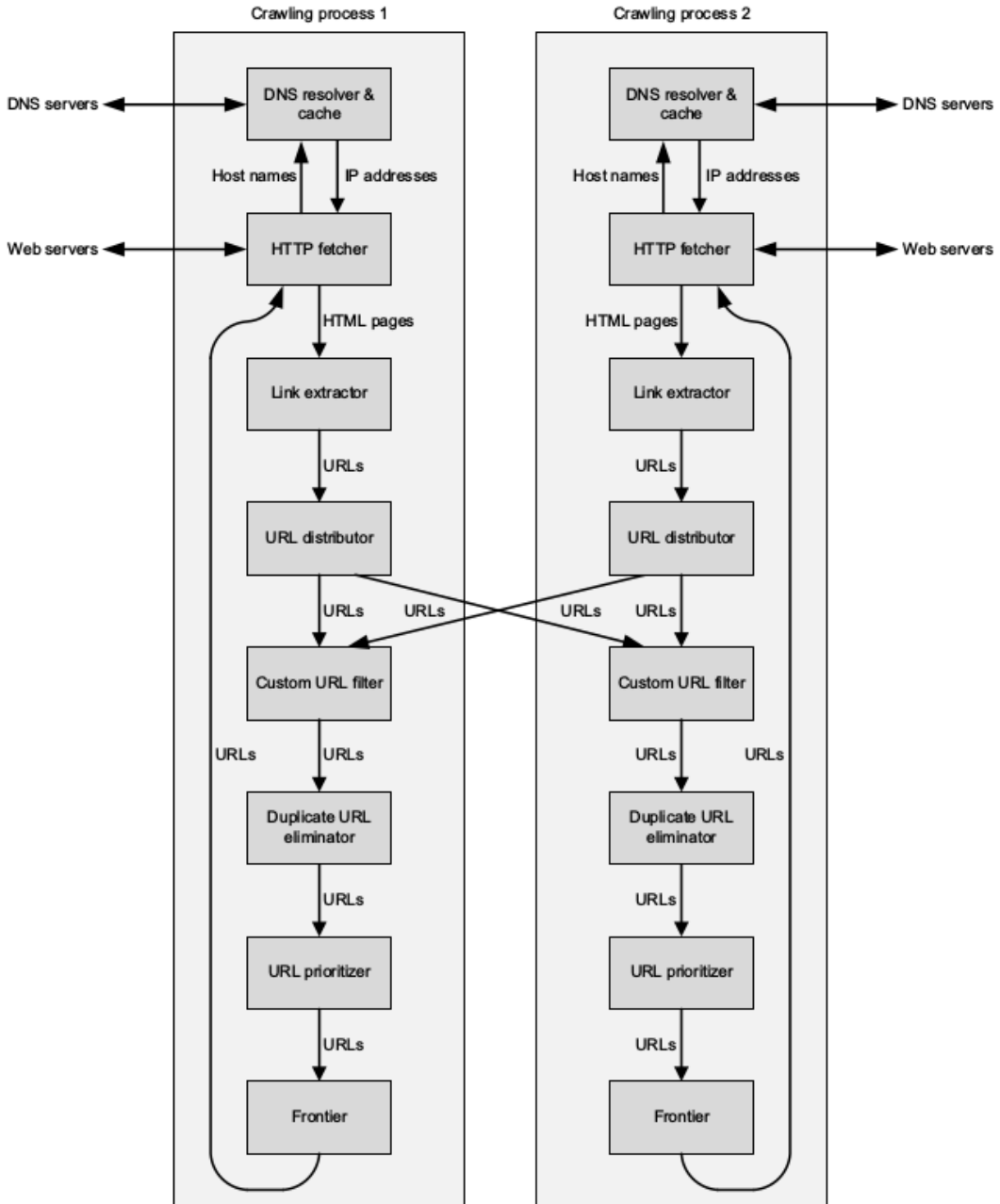
Супершинглы

- Для сокращения размерности вектора в min-hash применим алгоритм шинглов к итоговому вектору.
- Если хэш хотя бы одного супершингла для пары документов совпадает, то такую пару проверяем детальней.

SimHash

Основан на идее Locality-sensitive Hashing:

1. Вычисляем численный вектор с признаками.
2. Скалярно перемножаем с набором заданных случайных векторов.
3. Результирующий вектор переводим в скалярный хэш.


Crawling process 1
Crawling process 2
DNS servers
DNS resolver & cache
DNS servers
Host names
IP addresses
Host names
IP addresses
Web servers
HTTP fetcher
Web servers
HTML pages
HTML pages
Link extractor
Link extractor
URLs
URLs
URL distributor
URL distributor
URLs
URLs
URLs
URLs
Custom URL filter
Custom URL filter
URLs
URLs
URLs
URLs
Duplicate URL eliminator
Duplicate URL eliminator
URLs
URLs
URL prioritizer
URL prioritizer
URLs
URLs
Frontier
Frontier

Следующая лекция

Ссылочный граф Интернета