Chilkat: crawling

Marlon Dias
msdias@dcc.ufmg.br

Information Retrieval
DCC/UFMG - 2017

Introduction

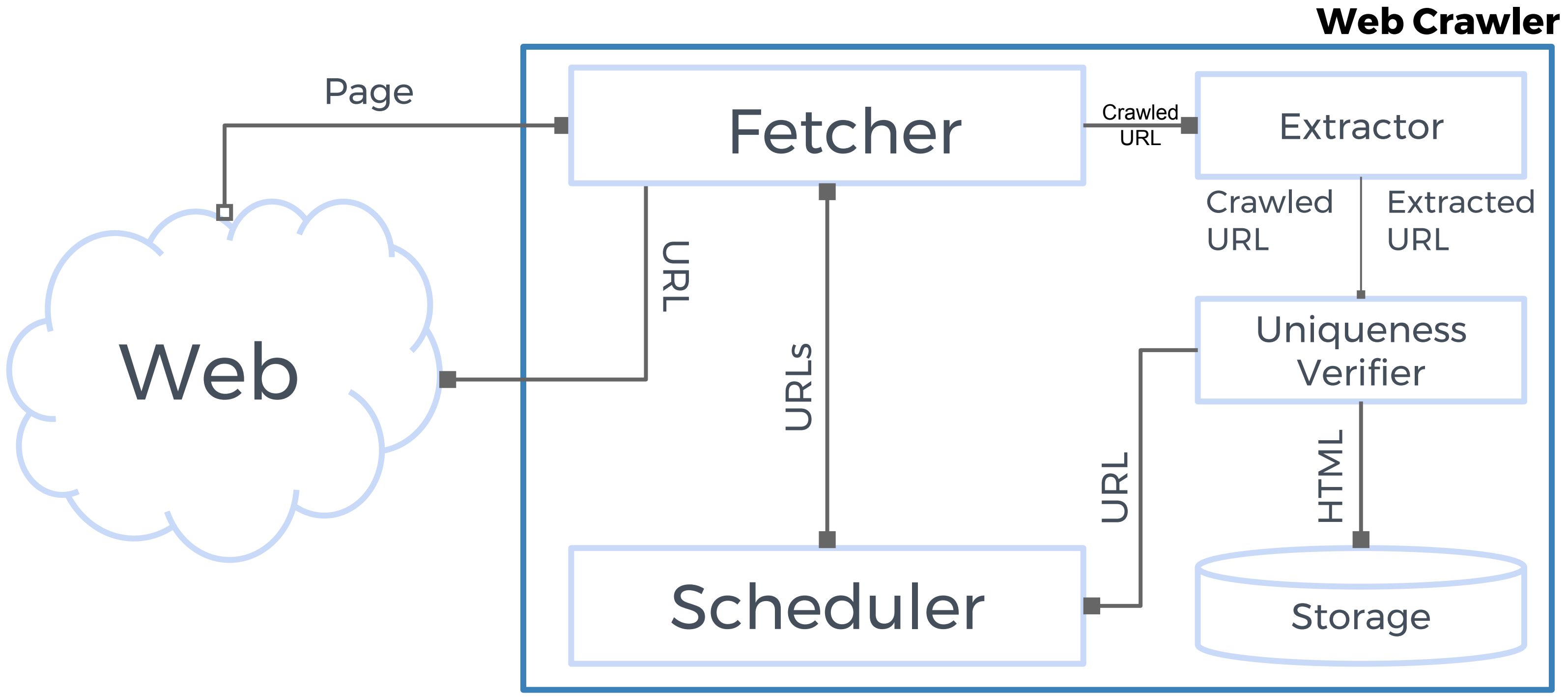
- Page collector (create collections)
- Navigate through links
- Unpleasant for some

Caution:

- Bandwidth
- Scalability
- Politeness
- Pages revisitation

Web Crawlers are also known as Web Spider, Web Robot or Bot

Introduction



Chilkat

Chilkat Spider - Web Crawler

https://www.chilkatsoft.com

Chilkat
lib

- C/C++ lib
- Crawl a web site.
- Accumulate outbound links for crawling other web sites.
- Robots.txt compliant.
- Fetch the HTML content of each page crawled.
- Able to crawl HTTPS pages.

https://www.chilkatsoft.com

Chilkat
lib

- Define "avoid" patterns to avoid URLs matching specific wildcard patterns.
- Define "avoid" patterns for avoiding matching outbound links.
- Read and connect timeouts.
- Maximum URL size to avoid ever-growing URLs.
- Maximum response size to avoid pages with very large or infinite content.
- Thread safe.

https://www.chilkatsoft.com

Chilkat
download

Linux

`https://www.chilkatsoft.com/chilkatLinux.asp`

Mac

`https://www.chilkatsoft.com/chilkatMacOSX.asp`

Chilkat
useful links

Examples

`https://www.example-code.com/cpp/spider.asp`

Documentation

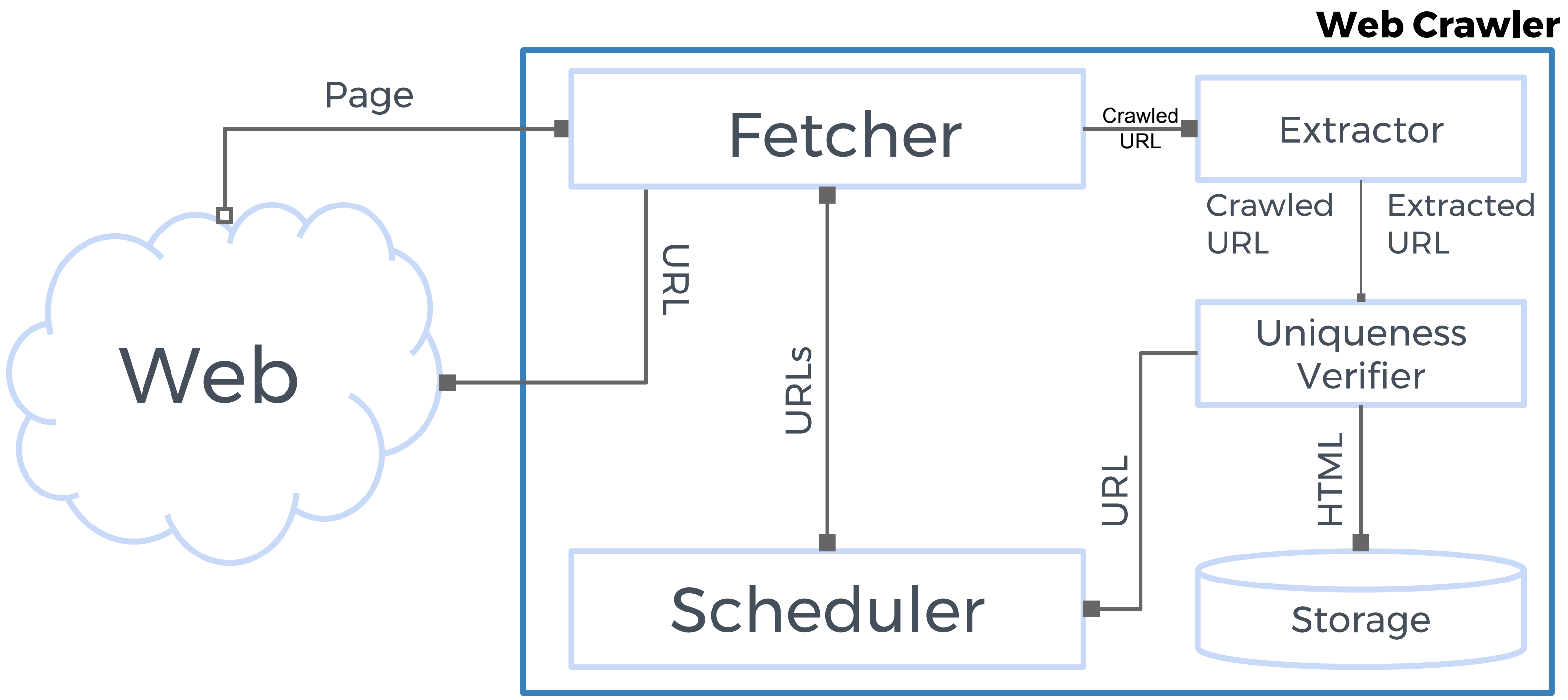
`https://www.chilkatsoft.com/refdoc/cpp.asp`

`https://www.chilkatsoft.com/refdoc/vcCkSpiderRef.html`

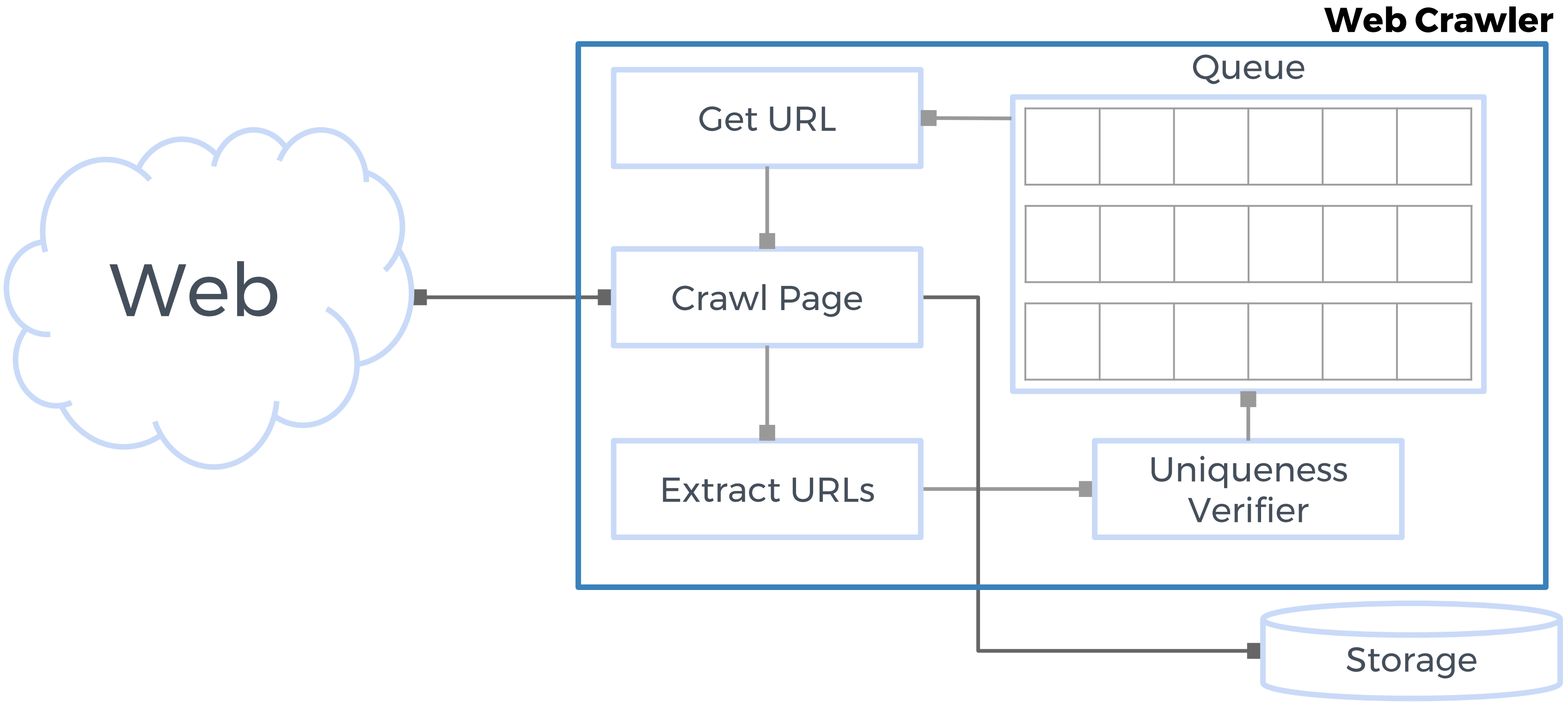
Building a crawler

Structure and Algorithm

Crawler
structure



Crawler
structure



Crawler
algorithm

```
CkSpider spider; string html, url; vector<string> queue;

spider.Initialize("www.chilkatsoft.com");
spider.AddUnspidered("http://www.chilkatsoft.com/");
spider.CrawlNext(); // bool return

html = spider.lastHtml(); // Saves HTML
int size = spider.get_NumUnspidered();
for (int i = 0; i < size; i++) {
     url = spider.getUnspideredUrl(0);
     spider.SkipUnspidered(0);
     queue.push_back(url);
}
```

Crawler
algorithm

- `spider.Initialize` called with just the domain name or a full URL
- If using only the domain name, URLs must be added to the unspidered list

```
spider.AddUnspidered
```

- Otherwise, the URL used in Initialize is the 1st URL in the unspidered list (domains don't work)

Crawler
algorithm

- `spider.Initialize` stays in a same domain
- It is a design project decision, probably helps in the way Chilkat works
- To change domains, one must initialize the spider again
  - or, is multiple spiders in different domains faster?

Crawler
algorithm

- Chilkat has **inbound** and **outbound** links
- Do the same for outbound links, no need to skip

```
    spider.getOutboundLink();
```

- Remove all outbound links at once

```
  spider.ClearOutboundLinks();
```

Crawler
algorithm

```
CkSpider spider; string html, url; vector<string> queue;

spider.Initialize("www.chilkatsoft.com");
spider.AddUnspidered("http://www.chilkatsoft.com/");
spider.CrawlNext(); // bool return

int size = spider.get_NumOutboundLinks();
for (int i = 0; i < size; i++) {
     url = spider.getOutboundLink(i);
     queue.push_back(url);
}
spider.ClearOutboundLinks();
```

Chilkat

Functionality

Chilkat
functionality

- Chilkat code is not open
    - we need some working around
- We need to use the statically linked libraries
    - `.so` for linux
    - `.a` for Mac
- They have to be in the same folder as your code

Crawler
makefile

```
UNAME_S := $(shell uname -s)
TOP := $(shell pwd)
ifeq ($(UNAME_S),Linux)
   FLAGS += $(TOP)/chilkat/lib/libchilkat-9.5.0.so
endif
ifeq ($(UNAME_S),Darwin)
   FLAGS += chilkat/lib/libchilkat.a
endif
```

Chilkat
functionality

- Chilkat has its own queue
    - `CrawlNext()` crawls the next URL queued
    - After `CrawlNext()` all inbound links are queued
- Local queue
    - Different techniques of scheduling may be applied
- Cleaning the queue is important

Chilkat functionality

- One may need a URL's domain
  - `getUrlDomain(url)` returns url's domains
- `getUnspideredUrl`: returns inbound links
- `getOutboundLink`: returns outbound links
- `get_NumUnspidered` and `get_NumOutboundLinks` inform number of links
- Chilkat has its own classes, such as `CkString`, feel free to use them

Designing a crawler

Your design

Designing a crawler

- Scheduling
  - a broad and horizontal exploration
- Web politeness
  - Robots
  - Not too many accesses
- Scalability
- Storage

Designing a crawler
scheduling

- Prioritize different domains
- Do not focus in the same site
    - Pay attention to the URL
- Design your own queue
    - a priority queue

```
http://www.ufmg.br = http://ufmg.br
http://dcc.ufmg.br/dcc = http://www.dcc.ufmg.br
http://www.dcc.ufmg.br/pos/
```

Designing a crawler
politeness

- **Do not get banned!**
- There are pages that do not want to be indexed, consequently crawled
    - respect the robots' file
- DDoS is difficult to detect
    - usually multiple accesses from the same IP address in short time indicates DDoS

Designing a crawler
scalability

- Crawling is simple
- There is not much processing
  - current processors can handle it!
- Main limitation is bandwidth
- Processor stay idle too long
- Multiple crawlers can keep it busy
  - Threads

Designing a crawler
storage

- Tons of html files will be collected
- Can the OS support that many files? How to handle them all later during indexing phase?
- Better put them all together, or as much as possible
  - the number of htmls in a file may vary

Designing a crawler
storage

- Since they will be together, better follow a pattern so one can retrieve them later
- For this project, we will respect the following pattern:

```
||| <url> | <html> ||| <url> | <html> |||
         ||| <url> | <html> |||
```

- Make sure **there is not a single** | (**pipe**) in the crawled html
    - one can edit the html and remove them

Evaluation

criteria

- Makefile is mandatory
    - or some sort of script to compile
- Code (30%):
    - Documentation
    - Code Structure
    - Web politeness
- Code must be compatible with Linux
    - test it using CRC PCs
        - SSH Linux | SSH Windows

Evaluation
criteria

- Documentation (30%):
  - Description of implemented crawler
  - Project decisions
  - Complexity analysis
- Experimental Results (40%):
  - Ex: # page collected, speed, bandwidth, quality, etc
  - Results analysis, pros and cons, improvements

Tips

Really important!

Tips

- I strongly recommend you to use C/C++
- Take advantage of C++
    - C++11 or C++14 may help you (even C++17)
- All default C/C++ libraries are allowed, therefore make the most of it, ex: threads, maps, time, etc
- **Mind the usage of memory!**
- Logging is important

Tips

- **Mind the web politeness**
    - Seriously, do not get banned!
- `'\n'` is faster than `std::endl`
- Testing is valuable
    - Save some time for experimenting

Tips

- Be careful with Chilkat
  - there is memory leak within Chilkat
    - ex: `CkString.split()`
- Report represents a big part of your grade
  - dedicate some time for writing
  - **LaTeX**