Crawling - Information Gathering on the World Wide Web

Kai Shen
Dept. of Computer Science, University of Rochester

Crawlers or Robots

- Definition: automated programs that fetch documents from Web sites
  - the complete or a significant portion of Web sites' document hierarchy
  - specific pages about a subject
- Usage:
  - search engines
  - comparison shopping
  - automated testing/server monitoring
  - or just for some fun

Manual fetch: telnet

```
[kshen@tallinn ~] telnet www.yahoo.com 80
Trying 64.58.76.176...
Escape character is '^]'.
GET /index.html HTTP/1.0
HTTP/1.1 200 OK
Date: Wed, 22 Jan 2003 21:37:16 GMT
Cache-Control: private
Connection: close
Content-Type: text/html
<html>
... ...
</html>
Connection closed by foreign host.
[kshen@tallinn ~]
```

Do it yourself

Open a TCP connection with **www.yahoo.com** at port 80
Send a TCP message with a HTTP request in it:
```
"GET /index.html HTTP/1.0\r\n\n"
```
(or with more header fields)
Wait for a response, parse the HTTP header in the response, make sure everything is correct
Show the fetched page content or store it in a disk file
Close the TCP connection
A nifty tool: wget

[kshen@tallinn ~]$ wget -O aaa http://www.yahoo.com
--18:05:16-- http://www.yahoo.com/
 => `aaa'
Resolving www.yahoo.com... done.
Connecting to www.yahoo.com[64.58.76.226]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]

[ <=>                                            ] 29,468   1.48M/s

18:05:16 (1.48 MB/s) - `aaa' saved [29468]

[kshen@tallinn ~]$ wget
can do a lot more.

How to crawl?

- start with a root set of URLs, then follow hyperlinks in the pages

Crawler: design 1.0 – the basic

Data structure
a queue/stack for yet-to-visit URLs

Control flow
dequeue a target URL
retrieve the page using HTTP
parse the page content, enqueue embedded hyperlinks
store the page to the disk

Crawler: design 2.0 – more efficient

Data structure
a queue/stack for yet-to-visit URLs

Multi-threaded control flow
dequeue a target URL
retrieve the page using HTTP
parse the page content, enqueue embedded hyperlinks
store the page to the disk
Crawler: design 3.0 - don't fetch any page more than once

Data structure:
- a queue/stack for yet-to-visit URLs
- a hashtable for visited URLs

Multi-threaded control flow:
- dequeue a target URL
- retrieve the page using HTTP
- parse the page content
- enqueue embedded hyperlinks not visited before
- store the page to the disk
- dequeue a target URL
- retrieve the page using HTTP
- parse the page content
- enqueue embedded hyperlinks not visited before
- store the page to the disk
- dequeue a target URL
- retrieve the page using HTTP
- parse the page content
- enqueue embedded hyperlinks not visited before
- store the page to the disk
- dequeue a target URL
- retrieve the page using HTTP
- parse the page content
- enqueue embedded hyperlinks not visited before
- store the page to the disk
- dequeue a target URL
- retrieve the page using HTTP
- parse the page content
- enqueue embedded hyperlinks not visited before
- store the page to the disk
- dequeue a target URL
- retrieve the page using HTTP
- parse the page content
- enqueue embedded hyperlinks not visited before
- store the page to the disk
- dequeue a target URL
- retrieve the page using HTTP
- parse the page content
- enqueue embedded hyperlinks not visited before
- store the page to the disk

deal with DNS aliases, ...

Performance analysis

- How to make it run faster?
- where is the performance bottleneck?
- What are the resources involved?
  - local CPU processing for TCP/HTTP protocol handling and the parsing of page content
  - local disk bandwidth
  - network bandwidth (local/remote)
- Assume average page size 5KB
  - raw power of a single CPU
    - several thousand fetches/second ⇒ around 10-30MB/s
  - disk bandwidth
    - up to 5MB/s (disk write)
  - network bandwidth from the Internet
    - a T1 link (1.5Mbit/s) costs around $1000/month

So how to make it run faster?

- Adding more CPU/disk bandwidth won't help
- The solution is to upgrade the network bandwidth; or
crawling from several sites

Targeted Crawling

- What if you only want pages about Sports?

- UR
- World Wide Web
- RIT

anchor text, or texts nearby
**Crawler/Robot Rules**

- Don't annoy anyone
  - Web sites are not designed for crawlers/robots
  - crawlers/robots consume resources at sites they visit
- Be accountable
  - Identify yourself (HTTP User-agent field)
  - Give a contact address (HTTP From field with your email address)
- Do it at opportune time, e.g. 3am-6am
- Don't visit anything the site owner doesn't want you to visit – robot exclusion (text file: /robot.txt)

```plaintext
http://www.cs.rochester.edu/robots.txt
User-agent: *
Disallow: /cgi-bin/
```

Take a look at http://www.cnn.com/robots.txt

---

**Control the fetch rate**

- **fetch N pages each second**
  
  - dequeue a target URL
  - retrieve the page using HTTP
  - parse the page content, enqueue embedded hyperlinks not visited before
  - store the page to the disk
  - start a new thread to fetch a page
  - sleep for 1/N second

```plaintext
you can't sleep arbitrarily short
```

---

**Control the byte rate**

- **fetch N bytes each second**
  
  - dequeue a target URL
  - retrieve the page using HTTP
  - parse the page content, enqueue embedded hyperlinks not visited before
  - store the page to the disk
  - check how we are doing so far
  - if we are under the target byte rate, start a new thread to fetch a page
  - sleep for a little

```plaintext
no guarantee on spikes
```
How to crawl 300M pages in a month?

- crawling from data centers of high-tier ISPs where you have fat pipes
- do it at night when you can pay less
- assume the mean page size is 5KB and we can crawl 10 hours a day - bandwidth 1.39MB/second
- data accumulation
  - we accumulate half a terabyte data every 10 days
  - how to transfer the data to your host site?

Assignment #1

- Implement a crawler
- I will give you a list of URLs to crawl so you don’t have to parse the page content to find embedded hyperlinks
- your crawler should be able to work in three modes:
  - as fast as it can
  - control the fetch rate
  - control the byte rate
- Don’t worry about the issues we haven’t resolved in class