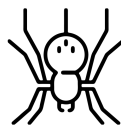


ALGORITMICA PER IL WEB

Prof. Sebastiano Vigna
6 CFU

Luca Cappelletti

Lecture Notes
Year 2017/2018



Magistrale Informatica
Università di Milano
Italy
October 10, 2017

Contents

1	Chapter	2
1.1	Robots.txt	2
1.2	Multi-threading	4
1.2.1	Design sincrónico	4
1.2.2	Design asincrono	4
1.3	Visit State (VS)	4
1.4	Problemi comuni	5

Chapter 1

Chapter

1.1 Robots.txt

Si tratta di un file standard, inserito nella root di siti web, con il quale è possibile inserire rule che i crawler devono rispettare. Solitamente quando una persona vuole crawlare un sito va a leggere e parsare questo file una volta all'ora.

```
1  # Notice: Crawling Facebook is prohibited unless you have express written
2  # permission. See: http://www.facebook.com/apps/site_scraping_tos_terms.php
3
4  User-agent: Applebot
5  Disallow: /ajax/
6  Disallow: /album.php
7  Disallow: /checkpoint/
8  Disallow: /contact_importer/
9  Disallow: /feeds/
10 Disallow: /file_download.php
11 Disallow: /hashtag/
12 Disallow: /l.php
13 Disallow: /live/
14 Disallow: /moments_app/
15 Disallow: /p.php
16 Disallow: /photo.php
17 Disallow: /photos.php
18 Disallow: /sharer/
19
20 User-agent: baiduspider
21 Disallow: /ajax/
22 Disallow: /album.php
23 Disallow: /checkpoint/
24 Disallow: /contact_importer/
25 Disallow: /feeds/
26 Disallow: /file_download.php
27 Disallow: /hashtag/
28 Disallow: /l.php
29 Disallow: /live/
30 Disallow: /moments_app/
31 Disallow: /p.php
32 Disallow: /photo.php
33 Disallow: /photos.php
```

```
34 Disallow: /sharer/
35
36 User-agent: Bingbot
37 Disallow: /ajax/
38 Disallow: /album.php
39 Disallow: /checkpoint/
40 Disallow: /contact_importer/
41 Disallow: /feeds/
42 Disallow: /file_download.php
43 Disallow: /hashtag/
44 Disallow: /l.php
45 Disallow: /live/
46 Disallow: /moments_app/
47 Disallow: /p.php
48 Disallow: /photo.php
49 Disallow: /photos.php
50 Disallow: /sharer/
51
52 User-agent: Googlebot
53 Disallow: /ajax/
54 Disallow: /album.php
55 Disallow: /checkpoint/
56 Disallow: /contact_importer/
57 Disallow: /feeds/
58 Disallow: /file_download.php
59 Disallow: /hashtag/
60 Disallow: /l.php
61 Disallow: /live/
62 Disallow: /moments_app/
63 Disallow: /p.php
64 Disallow: /photo.php
65 Disallow: /photos.php
66 Disallow: /sharer/
67
68 User-agent: Applebot
69 Allow: /ajax/pagelet/generic.php/PagePostsSectionPagelet
70 Allow: /safetycheck/
71
72 User-agent: baiduspider
73 Allow: /ajax/pagelet/generic.php/PagePostsSectionPagelet
74 Allow: /safetycheck/
75
76 User-agent: Bingbot
77 Allow: /ajax/pagelet/generic.php/PagePostsSectionPagelet
78 Allow: /safetycheck/
79
80 User-agent: Googlebot
81 Allow: /ajax/pagelet/generic.php/PagePostsSectionPagelet
82 Allow: /safetycheck/
83
84 User-agent: ia_archiver
85 Allow: /about/privacy
```

```

86 Allow: /ajax/pagelet/generic.php/PagePostsSectionPagelet
87 Allow: /full_data_use_policy
88 Allow: /legal/terms
89 Allow: /policy.php
90 Allow: /safetycheck/
91
92 User-agent: msnbot
93 Allow: /ajax/pagelet/generic.php/PagePostsSectionPagelet
94 Allow: /safetycheck/
95
96 User-agent: *
97 Disallow: /

```

1.2 Multi-threading

Siccome l'IO della scheda di rete è a 2-3 decadi di tempo superiore rispetto all'analisi effettiva dei dati, è necessario creare una grande parallelizzazione (anche 5000 thread), con **crivello** o struttura condivisa.

1.2.1 Design sincrono

Un approccio è creare un thread immaginandolo come solitario, quindi avviarne 10000 e affidare il lavoro di sincronizzazione al sistema operativo. Questa sarà la versione che andremo a realizzare in questo corso. Con questo approccio abbiamo i seguenti problemi:

1. Ogni thread deve lavorare su un unico url, altrimenti incontriamo problemi con i siti. Viene risolto con la **coda con delay**, che aspetta che l'elemento in cima alla coda abbia un timestamp inferiore al timestamp corrente e blocca il thread fino a che non riceve l'ok dalla coda.

1.2.2 Design asincrono

Un altro approccio è lavorare alla sincronizzazione tra i thread avviando tante richieste e lasciando lavorare i thread mano a mano che arrivano le risposte delle richieste.

È dibattuto quale dei due approcci sia il più veloce.

1.3 Visit State (VS)

Per ogni sito abbiamo una coda fifo di url ed un timestamp, e quando un thread incontra un timestamp minore di quello corrente acquisisce il **visit state** ed esegue la richiesta, scaricando alcuni degli url presenti alla lista connessa a quel nodo. Ad ogni visit state corrisponde un host, per cui nessun thread può fare richieste allo stesso host.

#	Timestamp	Host
0	1507621197	https://google.com
1	1507621097	https://facebook.com
i
n	1507601097	https://twitter.com

Figure 1.1: Coda di host del Visit State

#	Timestamp	URL
0	1507621197	https://google.com/search
1	1507601197	https://google.com/images
i
n	1504621197	https://google.com/maps

Figure 1.2: Coda degli url del primo Visit State

1.4 Problemi comuni

1. I server DNS centrali raramente reggono grossi carichi, per cui è necessario realizzare un **server ricorsivo locale**, per agente, separato. Le richieste a questo modo risultano apparire da IP diversi.
2. In aggiunta, per rispettare l'**IP politeness** è possibile realizzabile una coda di IP, contenente un timestamp che è il primo momento in cui si intende visitare quell'ip e un puntatore alla coda di visit state, di cui ogni nodo ha come priorità il valore pari al massimo tra il timestamp dell'ip e quello dei vhost.