Design, implementation and experiment of a *YeSQL* Web Crawler

{pierre.jourlin,romain.deveaud,eric.sanjuan}@univ-avignon.fr
{jeanmarc.francony,francoise.papa}@umrpacte.fr

Université d'Avignon et des Pays de Vaucluse

Introduction

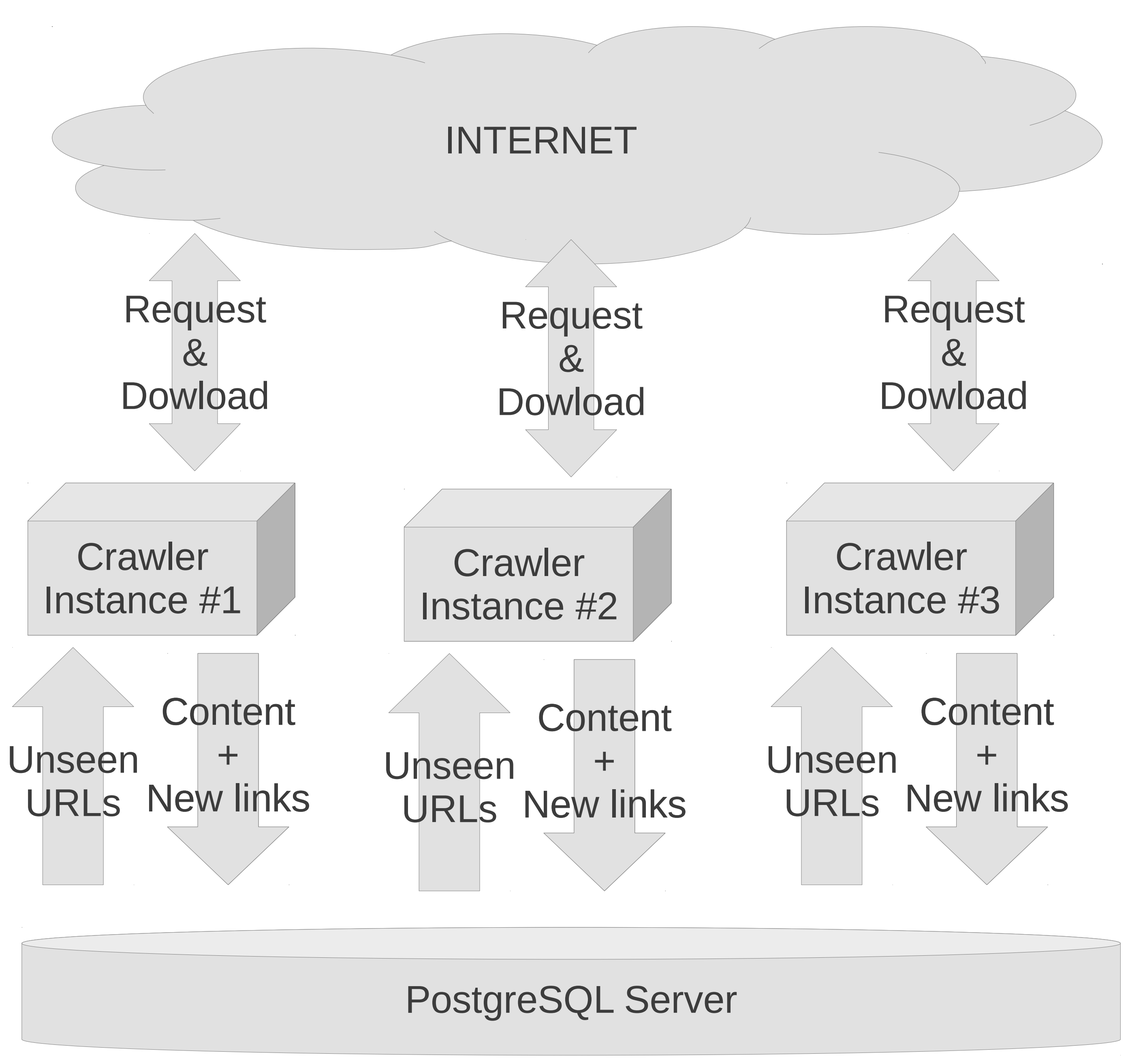
Where scalability is concerned, Apache Nutch®[a] and Heritrix[b] are probably the best-known and the most-accomplished open-source web crawlers.

However, the Java-language source code of these two software toolkits are rather large and complex.

We introduce a lightweight web crawler implemented in 911 lines of C and 200 lines of SQL and PL/SQL, that takes advantage of the capabilities of a PostGreSQL server [2].

[a] http://nutch.apache.org/
[b] https://webarchive.jira.com/browse/HER

Software Description



- Links and URLs' data are stored in a PostgreSQL[a] database.
- The user can launch several crawler's instances on several, possibly distant machines.
- Each instance of the crawler iteratively:
  1. fetches a list of URLs to be explored by sending a simple SQL query to the database;
  2. downloads the web pages;
  3. extracts new hypertext links to possibly new URLs;
  4. sends the new data back to the server.

[a] http://www.postgresql.org/

References

[1] J.-P. Cointet and C. Roth. Local networks, local topics: Structural and semantic proximity in blogspace. In *ICWSM '10*

[2] G. M. Roy. Perspectives on NoSQL. In *PGcon 2010: PostgreSQL Conference for Users and Developers*

Source Code

The source code is available under a GNU public license at `https://github.com/jourlin/WebCrawler`

Enhancing Crawling Strategies

Link or URL scoring can be easily expressed through PL/SQL programming and does not require advanced computer skills.

For example, the following function $s(a)$ attributes a score to a candidate anchor text a with respect to a predefined set of keywords $Q = \{k_1, ..., k_n\}$:

$$s(a) = \sum_{k \in Q} \#(k, a)$$

where $\#(k, a)$ is the number of times keyword k appears in the anchor text a.

This simple crawling strategy can be easily implemented as an SQL function:

```
CREATE OR REPLACE FUNCTION
ScoreLink(context text) RETURNS int AS
$$
DECLARE
score INT;
normcontext TEXT;
BEGIN
normcontext=normalize(context);
score=0;
IF (substring(normcontext, 'keyword1') IS NOT NULL) THEN
score = score +1;
END IF;
IF  (substring(normcontext, 'keyword2') IS NOT NULL) THEN
score = score +1;
END IF;
RETURN score;
END;
$$ LANGUAGE plpgsql;
```

The main advantage is that it does not require lots of computer skills. Hence people with a basic SQL knowledge can easily customize the crawling strategy so that it better fits their needs.

Use Case: Coverage of Tweeted URLs

Goal: validation of a political science hypothesis on the French presidential elections [1].

We used Twitter to measure the coverage between the URLs twitted by candidates official accounts and web pages crawled by our crawler.

Depth	# crawled URLs	% URLs covered (a)	% URLs covered (b)
0	2	0.00	0.00
1	34	0.08	1.00
2	1026	0.73	4.00
3	8543	1.84	8.00
4	56883	3.06	12.00
5	368247	7.33	27.00
6	2756671	15.28	40.00

Table 1: Tweeted URLs' coverage. (a): for all 4777 tweeted URLs ; (b): for the top 100 most frequently tweeted URLs. "Depth" is the minimum number of hyperlinks that one has to follow to reach an URL from the initial set.

These numbers are issued from a 1-week crawl that used only one crawler instance on a shared server.

Conclusion

Older and more ambitious projects such as Nutch and Heritrix certainly have lots of functionalities.

However this *YeSQL* web crawler can achieve a focused crawling which proved to be effective while searching for web sites related to the French presidential elections.

It is also efficient and easy to use and customize, which makes it an attractive tool for a wide community of researchers that may not be comfortable with computer science.