RDFaS

an RDFa Spider

Aim: to crawl the web extracting RDFa from pages

What is RDF and RDFa?

The Resource Description Framework (RDF) is a general method for the description or modeling of information that is implemented in web resources. RDF statements are represented as subject-predicate-object triples.

RDFa is an extension to XHTML markup which allows authors to turn existing human-visible test and links into machine-readable data without repeating the content. XHTML attributes are used to represent RDF triples.

The following HTML shows a heading with the content "Ivan Herman", which is the name of a person. However at face value, a computer would not be able to tell that this is a name.

```
<h1>Ivan Herman</h1>
```

An attribute can be set for the heading element, giving the content a property, in this case "foaf:name":

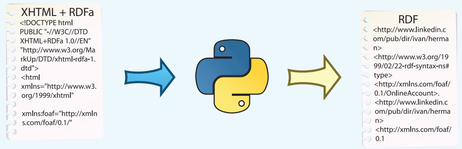
```
<h1 property="foaf:name">Ivan Herman</h1>
```

The prefix before the colon is the namespace and the suffix is the property. The foaf is linked to the foaf ontology, which is used to represent relationships between people and the name property indicates that the content represents the name of something.

Parsing Web Pages to extract RDF

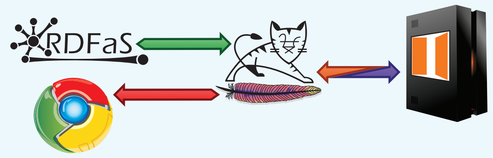
RDFa information when included in web pages is very bulky as not all of the extra XHTML from the page is needed. The ideal representation is as RDF triples which extracts the subject, predicate and object from the XHTML. Triple hierarchies can be inferred where the object of one triple matches the subject of another.

W3C's Distiller (also known as pyRDFa) is used for parsing.



Storing RDF Data - Triple Stores

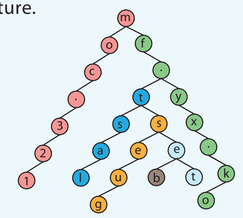
RDF data is conventionally stored in type of database system called a Triple Store. Some triple stores have a native storage backend tailored at storing RDF and performing reasoning, others use existing database engines like MySQL. Sesame, the Triple Store used for this project, provides access using Tomcat for adding and querying data over HTTP.



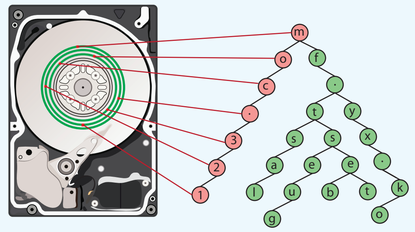
The URL Cache - Trie Implementation

The URL Cache analyzes URLs from the Crawler to determine whether or not they have been seen before and therefore whether or not they have been Crawled before or queued to be Crawled.

The cache is a large index of URLs, stored as a Trie (a form of tree structure). Duplicate prefixes are eliminated due to the hierarchial structure.



Unfortunately the Trie suffers from a performance flaw. Trie nodes are added sequentially as they occur from requests but often read in the order of the hierarchy meaning lots of drive head jumps (or seeks) will be required adding varying levels of overhead. On average a seek takes 10ms meaning a 100 character URL will take 1000ms (1 second) to check.



The URL Queue - Paged Queue

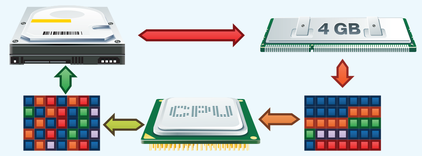
The URL Queue is a crucial part of the system, it tells the Crawler Threads the next URL to crawl but also determines the order in which URLs are crawled to avoid over-crawling any particular site.

URLs could come in to the queue in any order, it is likely many from the same site will be discovered in a short space of time.

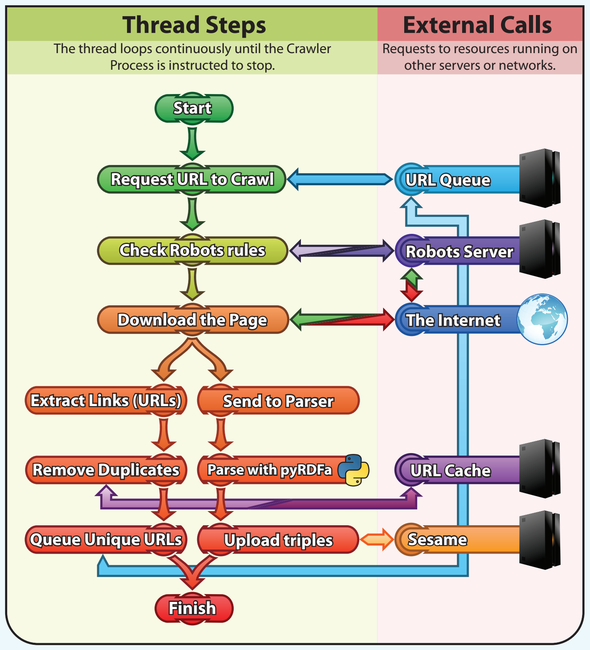
As URLs enter the queue they are stored in a Temporary Page until the page becomes full. Once it is full they are rearranged into a polite, non-sequential order (based on the domain) and saved in a Queue Page. Any domains that occur too frequently are put back in a Temporary Page to be thinned out during the next processing cycle.

The ideal diversity of a Queue Page (the limit on the most frequent domains) is automatically tuned based on recent crawling and speed statistics.

The queue is broken down in to pages so that data can be read and written in sequential chunks and each page is always small enough so that it can be processed in memory.



A Crawler Thread



The Crawler Process

Each Crawler Process by default runs 20 crawler threads continuously until it receives an interrupt signal, at which point it joins all threads after they finish their current loop.

All the crawlers make requests to common external services running on other servers and networks.

The Master Server

The server residing between the rest of the cluster and The Internet is the Master Server. It provides distributed service lookup, common settings, routing and DNS resolution.

The Crawl Cluster

24 Server Cluster

A 24 server cluster is used to run the Crawler System. Each server is installed with Ubuntu 9.10, GCC C++, Python and pyRDFa. They are connected by a 100MBit network. Data is stored on the MFS Distributed File System for redundancy.

The 20 Crawl Servers

Twenty of the available servers run the crawler process so there are usually 400 Crawl Threads running.

The URL Cache & URL Queue

The URL Queue manages URLs to be crawled and the URL cache identifies duplicate URLs so that they are not re-crawled. They require vast resources and run on the most powerful server.

The Robots Server

So that politeness rules can be adhered to the crawler must often check a site's robots file. The Robots Server caches this data to minimise bandwidth.

Sesame

Sesame is the Triple Store used to archive the RDF data extracted from web pages during crawls. It allows for queries and logical reasoning to be applied to the data.

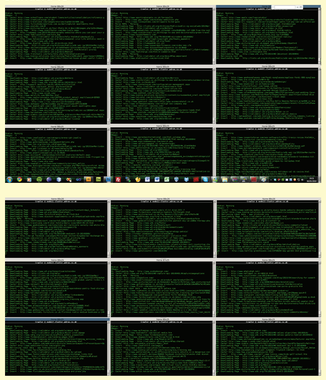
Monitoring the Crawl

The Importance of Monitoring

Crawling is a very delicate process, but crawling rapidly requires even more care and control. The Spider is capable of crawling at over 300 pages per second, if all these requests are directed at one server it could overload the server and deny access to legitimate users. Even though politeness rules should prevent this, errors and mistakes often exist so the Spider needs to be carefully observed, at least until it is proven to be stable and polite.

Screen Real-Estate

Almost 10 million pixels of screen space are needed to monitor all operations of the Spider - thats about 12 standard 15" screens.



Querying the Triple Store

Triple Stores can be queried in a simple manner to extract information or more complexly to run logical reasoning over the stored RDF triples.

A language called SPARQL which loosely resembles SQL has been developed for queries.

The following query requests the names and corresponding emails of people.

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
  ?person a foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:mbox ?email.
}
```

It is important to remember that the properties name and email only actually correspond to a name and email because of their definition in the foaf ontology.

RDFa Web Spider by Paul Ridgway
Supervised by Fabio Ciravegna