Incremental Crawling

Kevin S. McCurley
Google Research

SYNONYMS

spidering; crawler

DEFINITION

Part of the success of the World Wide Web arises from its lack of central control, because it allows every owner of a computer to contribute to a universally shared information space. The size and lack of central control presents a challenge for any global calculations that operate on the web as a distributed database. The scalability issue is typically handled by creating a central repository of web pages that is optimized for large-scale calculations. The process of creating this repository consists of maintaining a data structure of URLs to fetch, from which URLs are selected, the content is fetched, and the repository is updated. This process is called *crawling* or *spidering*.
Unfortunately, maintaining a consistent shadow repository is complicated by the dynamic and uncoordinated nature of the web. URLs are constantly being created or destroyed, and contents of URLs may change without notice. As a result, there will always be URLs for which the content is not present in the repository, as well as URLs whose content is different from the copy in the repository. Many new URLs can only be discovered by recrawling old URLs whose content has now changed to include links to new URLs. In order to minimize the impact of these inconsistencies, URLs should periodically be prioritized and revisited. The process of prioritizing and revisiting URLs is usually referred to as *incremental crawling*. The primary issues in incremental crawling center around defining metrics for performance, both for the quality of the repository and the resources required to build and maintain the repository.

HISTORICAL BACKGROUND

In the early days of the world wide web, it quickly became apparent that documents could be treated as living objects that could be changed at will. Thus the notion of a URL was born as a "resource locator" rather than a document ID. Early attempts to build search engines largely ignored this problem, assuming that if a document was worth retrieving later, then it would remain relatively stable at its URL. Thus early attempts to crawl and index the web simply started with a set of seed URLs, and iteratively crawled pages and extracted new URLs to be crawled. After a period of time, this repository would be used as a "snapshot" of the web to build a keyword index. If a reasonable seed set is used and URLs are appropriately prioritized, the resulting snapshot was useful for constructing a search engine or performing aggregate analysis.
The snapshot approach works well for web pages that are created and remain unchanged, but web usage has evolved quite a bit since the early days. Web authors learned that fresh content would bring repeat readers, and readers have increasingly migrated toward entertainment and news, for which freshness is increasingly important. Thus in some sense the more dynamic parts of the web are those that have the highest readership and therefore the greatest social impact. As a result, incremental crawling strategy has become increasingly important.

SCIENTIFIC FUNDAMENTALS

In order to address the issues around incremental crawling, we should first define the goals of the shadow repository. The primary application that has been implemented thus far have been search engines that allows users to find pages that match simply expressed information needs. A prerequisite for success of a search engine is that it

reference the best content that is of interest to a majority of users at any given time. For this, the repository should be as complete as possible and as fresh as possible, since pointing users at pages that don't fulfill their information need will result in a bad user experience.
Another application that can use a shadow repository is the discovery of plagarism or copyright-protected content on the web. A third application might seek to track historical evolution of the web, for historical studies. In this case the repository should not only contain the most recent content for a URL, but all versions as they evolve over time. It is easy to see that performance metrics for these applications will vary somewhat. Moreover, these applications will differ in the the degree of cooperation that can be expected from contributors to the web.

Metrics for incremental crawling There is no uniformly accepted notion of how to measure the inconsistency between the repository and the actual web. Moreover, even if there was universal agreement on the proper metric for consistency, it may be difficult to measure, and any such measure would be time-dependent since the web is constantly changing. Three possible metrics that are most obvious are:

> **Coverage:** count the number of pages that exist on the web but are not present in the respository at any given time.
>
> **Freshness:** count the number of documents that are present in the repository, but whose content was changed after insertion into the repository. Alternatively, the metric may incorporate a measure of the size of the change for individual pages.
>
> **Age:** the average age of documents in the repository (e.g., time since they were last updated).

For any particular application, these metrics may all be adjusted to incorporate weights of individual pages to reflect their relative importance for the application. Thus for a search engine, documents that are dominated by thousands of other documents for every possible query should probably should be weighted lower than documents that are often found by users of the search engine. The quality of an incremental crawling strategy must also be evaluated on the basis of resources that it consumes in order to maintain a given level of metric.

Incentives and cooperation It should be noted that a crawler consumes resources both of the party maintaining the repository as well as the creators of content on the web. If a crawler becomes too aggressive in tracking the changes to URLs, it could quickly overwhelm small sites and degrade the value of the web site to the users of the web. At the same time, it is relatively simple to create a web site that issues dynamic content in response to any request by a crawler, essentially creating an excess of URLs that could theoretically be fetched, archived, and indexed by the crawler. Web sites may also engage in other forms of mischief, such as accepting connections and either responding slowly or with noncomformant content.
Thus both content providers and repository maintainers are threatened by potentially unconstrained resource requirements. As the web has evolved, a number of standards and standard practices have emerged that mitigate these threats, and a form of economic equilibrium has emerged to allow repositories to be maintained.
The situation is complicated by the fact that the goals of content producers and crawlers are sometimes not aligned to each other. Content providers are generally motivated by the desire to shape public opinion through their information, or their desire to drive traffic for commerce. In a competitive market, multiple content providers may find themselves in competition with each other, competing for the attention of humans. As such, they can be expected to engage in a variety of practices that will improve their share of readership.
By contrast, consider the goals of a commercial search engine. Search engines make money from advertising, and in order to maintain the traffic, they need to serve the aggregated needs of users. Even if the search engine had a completely accurate copy of the web, there is still the task of deciding which results to show, and this is where the goals of individual content providers may conflict with those of the search engine (and users). Content providers generally want their content to be indexed, but they also want their content to be shown to users ahead of their competitors. They can be expected to act in their own self interest.
One of the activities that some sites engage in is to create link farms, which are designed to enhance the ranking of pages in search engines. In order for this to be effective, the pages of link farms must be crawled and incorporated into the ranking of the search engine. This is an extreme example of the more general phenomenon that content providers and search engines may disagree on what should be crawled and indexed.

Cache Consistency Inconsistency of a web shadow repository is similar to any other cache consistency problem, and has been studied extensively in the literature. Traditional approaches to cache consistency include time-to-live (TTL), client polling, and invalidation protocols. Most of the literature on caching has neglected the issue of discovery, which is a critical feature of crawling and one reason why pages need to be revisited (to find new links). Due to the uncoordinated nature of the web, most of the effort has gone into client polling approaches, although elements of the other approaches have also been experimented with. The HTTP protocol [5, section 13] contains a variety of optional features that can help with TTL approaches if they are properly implemented. Examples include the `Cache-Control`, `Expires`, `Last-Modified`, `If-Modified-Since`, `If-Unmodified-Since`, and `Vary` header fields. The cache-consistency protocol of the HTTP protocol is designed to facilitate human interactive browsing, and is not designed for batch processing. Moreover, relatively few web sites have taken the care to implement the existing protocols correctly.

In 2005 Google published an improved mechanism for conveying TTL cache consistency messages, known as sitemaps [4]. This followed an earlier effort called Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), and was released under a creative commons license. The purpose of the sitemaps initiative was to provide a more efficient polling mechanism, where web sites could provide information in an XML format to specify a list of valid URLs for crawling, along with optional last-modified dates, change frequencies, and priorities. Theoretically this scales up to sites with fifty million URLs, but has only weak consistency guarantees incorporated into it. In addition to conveying information about expiration and priority, the protocol also specifies a way for sites to notify individual search engines of the existence of their sitemap, either through placement in a robots.txt file or through a ping notification message [4]. Sitemaps can also be useful in eliminating intra-site duplicates, which consume resources of search engines and degrade the quality of indexing of sites.

In theory, sitemaps can greatly improve the efficiency of maintaining a shadow repository, but they have their own problems due to previously mentinoed misalignment of incentives between content providers and search engines. Sitemaps are subject to various kinds of abuse, and cannot be completely trusted by the crawler to give an accurate view of a web site.

Resource management The resources consumed by incremental crawling can be significant. In one study in 2002 [7], it was estimated that 40% of Internet traffic is due to web crawlers retrieving pages. Similar figures have been observed by others, though the impact is probably higher for low-traffic sites than it is for high-traffic sites. From the point of view of the crawler, it's not obvious how to optimally design a crawl strategy in order to achieve a given level of one of a metrics such as freshness.

One tempting strategy is to adjust the crawl frequency for a page in proportion to the rate of change for the page. Thus if a page changes freqently and substantially, it would receive a higher level of crawling. It is perhaps counterintuitive that this may not be optimal. In the extreme case, a page may in fact change every time it is accessed, which means that no matter how often you access it, you will never have the "up to date copy". Accessing a page too frequently would also violate the standard "politeness" policy of web crawlers.

Another potential policy would be to revisit pages with the same frequency, ignoring the change rate of individual pages. This is evidently wasteful of resources, but Cho and Garcia-Molina [1] showed both experimentally and theoretically that this outperforms the proportional approach. The optimal strategy interpolates between the two, with a visitation schedule that increases monotonically with the change rate, but penalizes pages that change too often.

Theoretical results of this type depend upon an underlying mathematical model of how the web changes, and should be carefully examined when applying it to a restricted subset of the web, and should be re-evaluated in the future should the social forces that shape the web somehow change. Such models should reflect the rate at which new information is produced, the distribution of such production among individual websites, and the rate at which information decays or is revised. Such models are essential for deciding how to balance the resources dedicated to discovery of new content vs. the confirmation that existing content has not changed.

KEY APPLICATIONS

To date, the primary application that has made use of crawling is search engines. The details of their incremental crawling strategies remain unpublished, as they are based on the perceived economic value of the repository. As the web has grown, search engines have become increasingly important for users, allowing them to express an

information need in fairly precise terms, and quickly navigate directly to documents on the topic of interest. From the point of view of a search engine, the dynamic nature of the web presents special problems, since a search engine will typically depend upon deep analysis of documents and links between documents. Documents whose content changes on a regular basis present a challenge for indexing, since the freshness of a search engine is a critical measure of utility for users.
Search engines are not the only applications that can make use of incremental crawling. Others include systems for prefetching proxies, notifications & alerts, mirroring and archiving, and business and political intelligence.

FUTURE DIRECTIONS

Most of the current strategies for incremental crawling are heavily dependent upon polling, and are therefore fairly inefficient. An approach built around notification and invalidation would clearly be more efficient, but there appears to be little economic incentive to implement them. Existing mathematical models for the growth and change rates of the web are fairly simplistic, treating all web pages as equal and failing to recognize the inherent organizational structure of web sites and the different purposes for which pages are created. Finally, as new applications emerge for web shadow repositories, we can expect that new requirements may emerge.

EXPERIMENTAL RESULTS

See [1].

CROSS REFERENCE

- •Web Crawler Architecture
- •Focused Web Crawling
- •Data broadcasting, caching, and replication

RECOMMENDED READING

Between 3 and 15 citations to important literature, e.g., in journals, conference proceedings, and websites.

[1] Junghoo Cho and Hector Garcia-Molina, *Effective page refresh policies for web crawlers*, ACM Transactions on Database Systems, **28, no. 4** (2003), pp 390–426. `http://oak.cs.ucla.edu/~cho/papers/cho-tods03.pdf`

[2] Jenny Edwards, Kevin S. McCurley, and John Tomlin, *An Adaptive Model for Optimizing Performance of an Incremental Web Crawler*, Proceedings of the Tenth International World Wide Web Conference, Hong Kong, 2001.

[3] E. G. Coffman, Jr., Zhen Liu, and Richard R. Weber, *Optimal Robot Scheduling for Web Search Engines*, Journal of Scheduling **1** (1998), 15–29.

[4] Sitemap protocol specification. `http://www.sitemaps.org/protocol.php`.

[5] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Mastinter, P. Leach, and T. Berners-Lee, *Hypertext Transfer Protocol - HTTP/1.1*, RFC 2616 `http://www.w3.org/Protocols/rfc2616/rfc2616.html`.

[6] Marios D. Dikaiakos, Athena Stassopoulou, and Loizos Papageorgiou, *An investigation of web crawler behavior: Characterization and metrics*, by Computer Communications 28 (2005), 880–897.

[7] X. Yuan, M. H. MacGregor, and J. Harms, *An efficient scheme to remove crawler traffic from the Internet*, Proceedings Eleventh International Conference on Computer Communications and Networks, Miami, 2002, pp. 90–95.

[8] Stefan Podlipnig and Laszlo Böszörmenyi, *A survey of Web cache replacement strategies*, ACM Computing Surveys **35, no. 4** (2003), pp 374–398. `http://doi.acm.org/10.1145/954339.954341`.

[9] Jia Wang, *A survey of web caching schemes for the Internet*, ACM SIGCOMM Computer Communication Review, **29, no. 5** (1999), pp. 36–46.