Cooperation schemes between a Web server and a Web search engine

June 11, 2003

Abstract

Search engines provide search results based on a large repository of pages downloaded by a web crawler from several servers. To provide best results, this repository must be kept as fresh as possible, but this can be difficult due to the large volume of pages involved and to the fact that polling is the only method for detecting changes.

In this paper, we explore and compare several alternatives for keeping fresh repositories that involve some degree of cooperation from servers.

1 Introduction

Modern Internet search engines play a key role by guiding users to relevant web pages related with their information needs. Today, it is estimated that web searches account for more than 10% of the total referrals to a web site [1], and more than 40% of new users [22]; and this figure is increasing every year.

Search engines keep an index based on pages downloaded from the web by a robot or crawler. As the number of publicly available web pages increases, the problem of keeping this index up-to-date with changes (in terms of updates, new pages and deletions) is more difficult [18], and usually an important proportion of the index is outdated [25]. This is bad for the search engine, but this is also bad for the website that is misrepresented, if we consider the whole user experience. For example, if a user searches for a word on a search engine and then goes to a page from the search results in `www.example.com` that no longer exists, or that contains material that is currently irrelevant for that user's information need, the user will be frustrated with both the search engine *and the web site* for not finding this information. Also, there is an opportunity cost related to this visitor: maybe the information he wants was moved to another page in the same website and the search engine was not aware of the change.

Another motivation for cooperation is that web crawlers can use an important amount of network and processor resources from web servers, specially if they don't follow existing rules of "good behavior" [15]. Web crawlers tend to visit many more pages than humans, and they request them very fast, normally with 10 to 30 seconds between visits; so they are believed to account for at least 16% of the requests [20]. Many of the requests are to unmodified resources, and can be avoided if the server cooperates with the crawler.

There are several things that a webmaster can do to improve the representation of a web site on the search engine's index and to prevent unnecessary visits from crawlers. In this paper, we show some existing techniques and propose new ones; some of the techniques shown were not designed for this specific purpose but can be adapted. We also present a framework for estimating and comparing the costs and benefits of each one.

The next section outlines previous work in this area, Section 3 presents the cooperation schemes and Section 4 is about how to evaluate them in terms of cost. Section 5 presents some concluding remarks.

2 Previous Work

In this paper, we only consider the cooperation between web servers and crawlers, not between crawlers: this issue is studied in [19], using a crawler simulator and proving that crawlers can benefit from sharing information about last-modification date of pages. In this case, the cooperation between search engines occurs at crawling time, but search engines could also exchange information later, like in the "STARTS" proposal [11].

There are several methods for keeping mirrors (replicas) of information services; these methods are not directly suitable for web server cooperation because the crawler usually is interested only in a subset of the pages (the most interesting ones) and not in the entire site. Mirroring methods include RSYNC [27], that generates a series of fingerprints for "chunks" of data, and then compares those fingerprints to compress and send only the modified parts, and CTM [13] that is a method for sending differences via e-mail, used to keep up-to-date copies of the source code of the Open BSD operating system.

A specific proposal for pushing last-modification data to web crawlers is presented in [12], including a cost model in which the meta-data is sent only if the web server is expected to be misrepresented above a certain threshold in the search engine. A more general Internet notification system was presented in [7].

The Distribution and Replication Protocol (DRP) [28] provides a protocol to distribute data using HTTP and data fingerprinting and index files. Another proposal that uses a series of files containing descriptions of web pages, is presented in [6].

Extensions to HTTP that use differences of content and compression to significantly reduce the total transfer time of web pages when using proxies were analyzed in [21], giving details on how to implement these extensions in practice.

DASL [24], the WebDAV searching and locating protocol, is a proposed extension to WebDAV [30] (Web-based Distributed Authoring and Versioning) that will allow searching the web server using an HTTP query with certain extensions, but neither the query syntax nor the query semantics are specified by the protocol.

3 Cooperation schemes

The standard HTTP transaction in which a web crawler fetches a page from a web server is shown in Figure 1. Note that meta data (information about the page) is downloaded along with the data.

The cooperation schemes we are going to study in this paper can be divided in two main groups: *interrupt* and *polling*.

In the *interrupt* (or *push*) schemes, the web server begins a transaction with the search engine whenever it is necessary; this can be when one or multiple pages are updated, based on server policies. The search engines must subscribe to the servers from which they want to receive events. This is similar to the relationship between the main processor and a hardware device (network card, scanner, etc.) in a modern computer.

In the *polling* (or *pull*) schemes, the search engine periodically requests data from the web server, based on search engine policies. The requests try to detect changes in the server and most crawlers

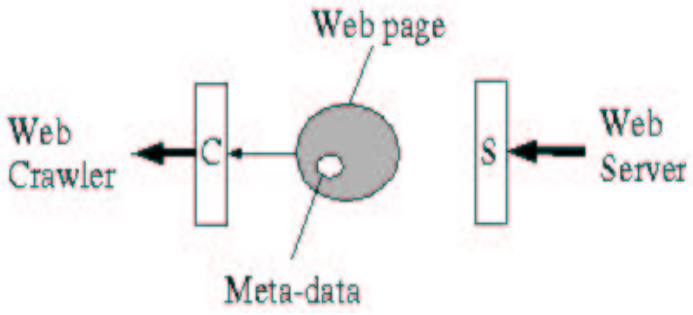

Figure 1: Without cooperation, the web crawler issues a request and then downloads the web page and the meta-data together.

use this method. Note that polling becomes equivalent to an interrupt when the polling period tends to zero; but the usage of resources at both ends of the polling line increase at the same time.

The schemes studied in this paper are summarized in Table 1.

Transfered data	Interrupt version	Polling version
Meta-data	Notify updates	Serve meta-data
Differences	Send differences	Serve differences
Pages	Send changed	Serve if-modified
Batches	Send batch	Serve multi-page request
Site	Send entire site	Serve entire site
Mixed	Remote agent	Filtering interface

Table 1: Summary of the cooperation schemes analyzed in this paper. All of them have two versions: interrupt (push) and polling (pull).

Before we get into the details of each scheme, there are some issues we must mention that are, in some sense, orthogonal to the scheme used:

Compression can be used for decreasing transmission cost at the expense of using more processing power on the server side. The possibility of requesting compressed responses to requests is considered in the HTTP protocol using the `accept-encoding` header.

Privacy issues arise when the crawler has access to information in the server that was not meant to be public. This may sound strange, but in practice when using a web crawler it is not uncommon to download files that were linked by mistake; we have even found complete plain-text password files!. Not all files in the web server public directories can be considered public, so most web server administrators are very reluctant to provide access for clients to list the content of directories.

Index update capabilities are very reduced in global-scale web search engines: constraints in terms of disk space are the most important limitation, so it is not always possible to store a complete copy of the downloaded pages. This can lead to some difficulties; for instance, removing a page or updating a paragraph without having the complete text can be very time-consuming in a standard inverted index. Also, in many cases updating batches of pages is preferred to updating single pages to reduce the overall cost.

Search engine "spamming" occurs whenever web server administrators try to get undeserved high ratings in the search engines. Data provided by web servers can't be trusted completely, for instance, in terms of page updates or page importance. Web crawlers used by most search engines

are interested only in some of the pages of a website, and the decission of which pages to add to the index must be left to the search engine.

Structure and HTML markup used in a website affects the visibility of its pages by search engine crawlers. Information that is accessible only through forms is, in general, impossible to gather for web crawlers; this is called the "hidden web" [23].

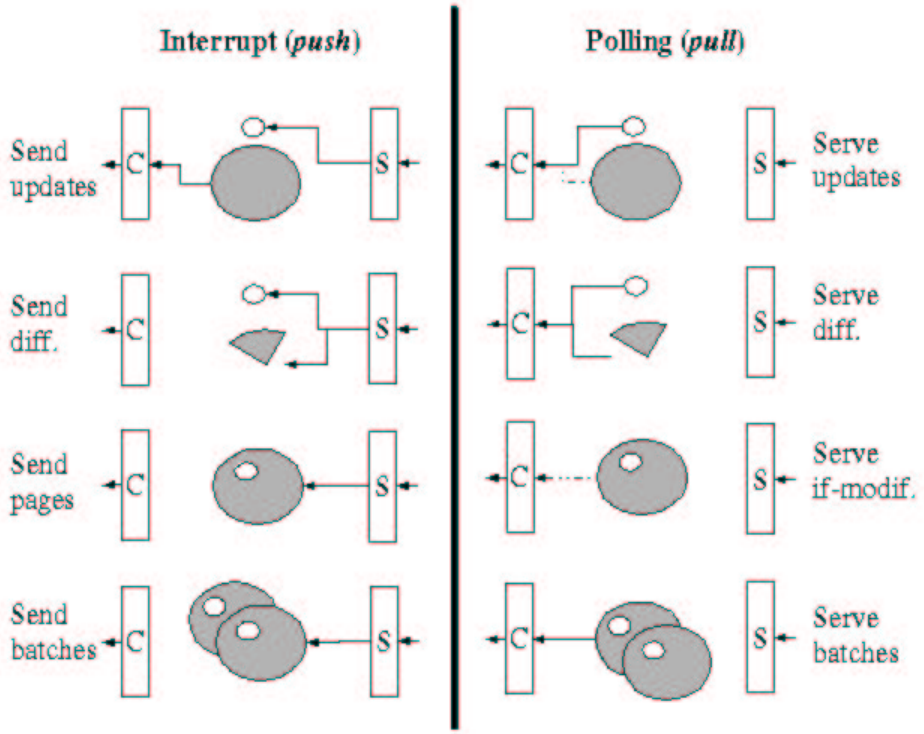

Figure 2: Diagrams of several schemes of cooperation. The arrow between the Web crawler "C" and the Web server "S" represents who initiates the connection. The small, white circle represents meta-data and the large, gray circle represents the contents of the page.

3.1 Interruption-based cooperation

In all these cases, the web server initiates a transaction with the search engine when needed. The search engine must subscribe with the web server to start receiving these notifications.

Send meta-data of updates: The propagation of the update is sent whenever there is an update on the web server (page change, deletion or new page), but the web server can wait and send the updates in batches. When receiving the meta-data about the update, the search engine may download, enqueue or ignore the event. Examples: the Keryx notification service, developed in the apogee of push-based content delivery in 1997 [7], Fresh Flow [12].

Send differences of content: Whenever an event happens, the web server sends a file containing the difference between the last version and the current one (if the change is big, then the web server may send the entire file). The search engine may accept or ignore the file received, but if it ignores it, then it must download a complete copy of the page to be able to receive differences for the following changes. Examples: CTM [13]; in this case, the difference is sent by electronic mail and the receivers execute the `patch` [29] program to apply the update to the local copy.

Strategy	Network cost	Processing (server)	Processing (crawler)	Freshness improvement
Send meta-data of updates	+	+		High
Send differences of content	– –	+ +	+	High
Send changed pages	–	+		High
Send batch update	+	+		High
Send entire site in one file	+ +	+		High
Remote agent	– –	+ +	–	High
Serve meta-data of updates	+	+		Normal
Serve differences of content	– –	+ +	+	Normal
Serve pages only if modified	–			Normal
Serve many pages in one request	–			Normal
Serve entire site in one file	+ +	+		Normal
Filtering interface	– –	+ +	–	High

Table 2: Relative costs of server cooperation schemes analyzed in this paper, against the base case where no cooperation exists: + means more cost, – means less cost. The last column is the expected improvement in freshness for the search engine collection.

Send changed pages. The web server sends the complete text of all updated pages or new pages when modifications are made. Examples: this was typical in push technologies [14], and was implemented in services like "Marimba Castanet" and "Pointcast" in the early days of the web. Currently, it is being used in some wireless devices [8].

Send multi-pages updates. The web server sends batches of modified pages according to a schedule defined by the web server. This can be useful if the updates are regular and involve several pages; for example, in the website of a daily or weekly newspaper.

Send entire site. The web server sends the entire site. This is useful, for instance, for uploading an entire web site when the site is publicly available for the first time, or if the web server changes a significant part of the site, as an extension to the previous scheme.

Remote agent. The web server executes software provided by the search engine; this software includes instructions to identify important pages for the search engine, and to detect changes in those pages that are relevant for the search engine. Important pages can be identified based on local connectivity or textual information, and changes can be detected using a custom algorithm that will depend on the search engine characteristics. When there is a change, the agent sends back some data to the search engine; this can be meta-data, complete pages or differences. This is a typical application for a mobile agent [16], and this cooperation can be taken one step further, as in some cases this software can help the search engine to fetch data from "near" servers, as proposed in [26].

3.2 Polling-based cooperation

Serve meta-data of updates. In this case, a file containing last-modification data -and probably file size- is served. To be useful, this file should contain a description of many pages on the web site. In the case of single files, the HTTP protocol provides `HEAD` requests that are responded with meta-data about the requested object. Multiple HTTP `HEAD` requests can be pipelined, but this is not as efficient as serving a concise file that refers only to the changed. Examples: the

distribution and replication protocol (DRP) [28]. In [6], files containing information about changes are served. In RDF [17], there is the possibility of informing time-related data about the resources. In WebDAV [30], the `PROPFIND` method allows to query for properties of documents, and the proposed `BPROPFIND` method allows to query for properties of groups of documents. Also, the HTTP `Expires:` header presents a way of informing the crawler of the next change in a web page, but this involves prediction and therefore is not widely used.

Serve differences of content. The web server provides a series of differences between a base version and newer versions. In the most simple case, the difference is only between the last and the current version. Examples: HTTP Delta responses proposed in [21] that use the `content-encoding` field of HTTP responses; also in many cases, source code for popular free software can be updated using the `patch` [29] program, with servers providing differences between the original version and the new version. For web sites, differences in terms of structural changes in the links, can be encoded using tables as in *WHOWEDA* [5].

Serve pages only if modified. The web crawler can avoid downloading a file if it has not been modified. This is done using a date that the crawler provides -usually the last visit to the same page- in HTTP/1.0. A more precise alternative is to provide an *entity-tag* (E-Tag): a *fingerprint* identifying the text of the document, as in HTTP/1.1. Examples: HTTP/1.1 [10] If-Modified-Since headers, used by a minority of crawlers [6], but supported by most web servers.

Serve multiple pages on one request. The overhead associated with multiple requests can be avoided by requesting a series of pages in the same request. Examples: this is usual for modern web browsers, and is implemented using an HTTP/1.1 [10] `Connection: keep-alive` header; in this case, pipelining of the requests can also be used.

Serve entire site. As latency is a very important component in web page transfers, specially in the case of small files, in some cases it is better to serve the entire site, or a significant portion, in one request. This is suitable only if the changes occur in many pages. Examples: typically, Linux distributions are distributed in whole CD-sized images and not on a file-by-file basis.

Filtering interfaces. In these cases, the web servers provide a standard method of answering queries from the crawler. The typical query a web crawler can ask is "give me all the pages that have changed since this date". A more powerful filtering interface could also include requests for differences, or querying about other characteristics such as page size or local importance. Examples: DASL for searching web servers [24], RSYNC [27] for mirroring content, CIP (Common Indexing Protocol) [3, 4, 2]; a general framework for these kind of filtering interfaces is in terms of Web Services [9].

The methods we have shown can be described with diagrams as in Figure 2 (remote agents and filtering interfaces are not shown).

4 Cost analysis

4.1 Costs for the web server

We will consider unitary (per-page) costs and benefits.

- b: Benefit for the web server from a user viewing one page.
- c_n: Network cost for serving one page, i.e.: bandwidth cost.
- c_p: Processing cost for serving one page, i.e.: servers cost.

A simple observation is that we should have (in theory) $b \geq c_n + c_p$, otherwise, the server would not be able to pay the operation costs. However, we should note that some web sites can be financed by revenues from other sources. Another observation is that in general processing capacity has grown faster than network connectivity in recent years; in general $c_n > c_p$.

Estimates: We cannot measure these quantities, but we can make some estimates: as of June 2003, the cost-per-click of an advertising campaign on the Web is about US$ 0.05, so probably $b \geq 0.05$. On the other end having a web server costs about US$ 10 for 5 gigabits of traffic, or 625Mb; if each page is 40Kb on average, it is enough for 15,000 page-views; notice that network bandwidth is usually "overbooked" in popular virtual servers, probably by a factor or 2 or 3, so an estimate of the cost is: $c_n + c_p \leq 0.002$.

This means that if we only account for web server usage, serving a web page costs at most 1/25 of the benefit, and this is probably the main explanation for the huge success of the world wide web as a platform for publishing information. The main source of cost when keeping a large website is not the web hosting, but rather the cost of producing and maintaining the contents.

In Table 2 we provide a rough estimation of relative costs associated with these cooperation schemes. Network bandwidth savings are the product of not dealing with unnecessary requests from the crawlers, and costs, from sending more than is necessary. Processing costs involve keeping meta-data, calculating differences, or more complex processing. Benefits arise from increased freshness on the web search engine, and are higher if an interruption (*push*) is involved.

Which is the best strategy for the web server ? This will depend on the price the web server is willing to pay; if this is minor, then using server software that correctly implements HTTP/1.1 is the best option. If the price is moderate, then serving and sending meta-data of updates is the best option. If the server wishes to invest more resources, it can benefit from providing content differences and/or a filtering interface for the crawlers.

4.2 Costs for the crawler

The main costs for the crawler for each page are: polling or receiving and interrupt, downloading and processing pages. We will consider that in terms of network and processing, generating a connection or handling an interrupt have the same cost.

The freshness of the repository is higher in interrupt-based strategies -there is no significant delay between the server update and the search engine syncing of the page. The costs for the crawler are summarized in Table 2. Network cost for the crawler is the same as for the server, as each transfer in these schemes involves both a crawler and a server.

Which is the best strategy for the crawler ? A remote agent or filtering interface can help to distribute the workload of the search engine, specially if servers cooperate in pre-processing documents or in generating partial indexes. The remote agent can be used for the more important websites (such as news sources) if the crawler can process the interrupts as they arrive, probably keeping a small index for the most changing data.

The extreme case of using an agent is when the web server generates the (partial) inverted index and then sends it to the search engine, that only needs to perform a merge. In this case, the crawling problem is simplified, and turns into polling or pushing of indexes.

4.3 Overall cost

It is very important to consider that not all web servers are equal, and the distribution of "quality" on the web is, by all measures, very skewed: most of the important web pages are on a few web servers, so the search engine should try to get larger web servers to cooperate.

By inspecting Table 2, a noticeable fact is that the schemes that do not require extra cost for the web server are already implemented in HTTP (`keep-alive` and `if-modified-since headers`).

Pushing or pulling differences of content are probably the most balanced schemes, because the server gains in less bandwidth usage. These schemes are more useful if many clients can benefit from differences -not only the web crawlers of search engines, but also the general public using enabled browsers or cache services.

It is clear to see that if the request-response paradigm is enforced strictly, then the only scheme that can provide high benefits in terms of freshness is a filtering interface.

5 Concluding remarks

How probable is the wide adoption of cooperation strategies ? The basic protocols are not completely implemented in the same way across different servers, and the minimum-common-denominator is quite poor in terms of functionalities; for instance: many web servers provide bad timestamps for the pages.

On the other hand, web site administrators can provide cooperation if it is not so costly and means an important benefit for them. This benefit should come mainly in terms of exposition on the web search engines. We consider that the reductions on load for the web server are probably not enough by themselves to justify the adaption of a cooperation strategy.

As of June 2003, the Open Directory Project lists more than a thousand firms that provide the service of "search engine optimization" for websites. This evidences interest from web site administrators to invest resources in this subject; we hope that this communication is a step to motivate the design of cooperation schemes that can gather critical mass in terms of support.

With the emergence of web services, filtering strategies could be an interesting possibility for the near future, as they can help crawlers and other autonomous agents to interact with web servers at a more meaningful level, bringing the networked nature of the web one step further.

References

[1] Search engine referrals nearly double worldwide. www.websidestory.com/pressroom/pressreleases.html?id= 2003.

[2] J. Allen, P. Leach, and R. Hedberg. RFC 2653: CIP transport protocols. www.ietf.org/rfc/rfc2653.txt, 1999.

[3] J. Allen and M. Mealling. RFC 2651: The architecture of the Common Indexing Protocol. www.ietf.org/rfc/rfc2651.txt, 1999.

[4] J. Allen and M. Mealling. RFC 2652: MIME objects definitions for the Common Indexing Protocol. www.ietf.org/rfc/rfc2652.txt, 1999.

[5] S. Bhowmick, S. K. Madria, and W. K. Ng. Detecting and representing relevant web deltas in WHOWEDA. *IEEE Transactions on Knowledge and Data Engineering*, (2):423–441, 2003.

[6] O. Brandman, J. Cho, H. Garcia-Molina, and N. Shivakumar. Crawler-friendly web servers. In *Proceedings of the Workshop on Performance and Architecture of Web Servers (PAWS)*, Santa Clara, California, 2000.

[7] S. Brandt and A. Kristensen. Web push as an Internet Notification Service. In *W3C workshop on push technology*, 1997.

[8] B. Charny. Wireless web embraces "push", 2002.

[9] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. WSDL: Web services description language. www.w3.org/TR/wsdl, 2001.

[10] J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. HTTP/1.1 - Hypertext Transfer Protocol. w3.org/Protocols/rfc2616/rfc2616.html, 1999.

[11] L. Gravano, K. C.-C. Chang, H. Garcia-Molina, and A. Paepcke. STARTS: Stanford proposal for internet meta-searching. In J. Peckham, editor, *Proceedings of International Conference on Management of Data (SIGMOD)*, pages 207–218. ACM Press, 1997.

[12] V. Gupta and R. H. Campbell. Internet search engine freshness by web server help. In *Proceedings of the Symposium on Internet Applications (SAINT)*, pages 113–119, 2001.

[13] P.-H. Kamp. OpenBSD CTM. www.openbsd.org/ctm.html, 2003.

[14] T. Kapyla, I. Niemi, , and A. Lehtola. Towards an accessible web by applying push technology. In *4th ERCIM Workshop on "User Interfaces for All"*, 1998.

[15] M. Koster. Robots in the web: threat or treat ? In *ConneXions, 4(4)*, 1999.

[16] D. B. Lange and M. Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, 1999.

[17] O. Lassila and R. Swick. World wide web consortium - rdf. www.w3.org/TR/REC-rdf-syntax, 1999.

[18] S. Lawrence and C. L. Giles. Accessibility of information on the web. *Nature*, 400(6740):107–109, 1999.

[19] G. L. McLearn. Autonomous cooperating web crawlers, 2002.

[20] D. Menasce, V. Almeida, R. Riedi, F. Pelegrinelli, R. Fonseca, and W. M. Jr. In search of invariants for e-business workloads. In *Second ACM Conference on Electronic Commerce*, 2000.

[21] J. C. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy. Potential benefits of delta encoding and data compression for HTTP. In *SIGCOMM*, pages 181–194, 1997.

[22] J. Nielsen. Statistics for traffic referred by search engines and navigation directories to useit. www.useit.com/about/searchreferrals.html, 2003.

[23] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *Proceedings of the Twenty-seventh International Conference on Very Large Databases*, 2001.

[24] J. Reschke, S. Reddy, J. Davis, and A. Babich. DASL - dav searching and locating protocol. www.webdav.org/dasl/, 2002.

[25] D. Sullivan and C. Sherman. Search Engine Watch. www.searchenginewatch.com/reports/, 2002.

[26] W. Theilmann and K. Rothermel. Maintaining specialized search engines through mobile filter agents. In M. Klusch, O. Shehory, and G. Weiß, editors, *Proc. 3rd International Workshop on Cooperative Information Agents (CIA'99)*, pages 197–208, Uppsala, Sweden, 1999. Springer-Verlag: Heidelberg, Germany.

[27] A. Tridgell and M. Pool. RSYNC: fast incremental file transfer. samba.anu.edu.au/rsync/, 2003.

[28] A. van Hoff, J. Giannandrea, M. Hapner, S. Carter, and M. Medin. DRP - distribution and replication protocol. www.w3.org/TR/NOTE-drp, 1997.

[29] L. Wall, P. Eggert, W. Davison, and D. MacKenzie. GNU patch. www.gnu.org/software/patch/patch.html, 2000.

[30] E. J. Whitehead, Jr. and Y. Y. Goland. WebDAV: A network protocol for remote collaborative authoring on the web. In *Proc. of the Sixth European Conf. on Computer Supported Cooperative Work (ECSCW'99), Copenhagen, Denmark, September 12-16, 1999*, pages 291–310.