Turning The Web Into An Effective Knowledge Repository

Luis Veiga
INESC-ID / IST
Rua Alves Redol, 9, 1000 Lisboa, Portugal
Email: luis.veiga@inesc-id.pt

Paulo Ferreira
INESC-ID / IST
Rua Alves Redol, 9, 1000 Lisboa, Portugal
Email: paulo.ferreira@inesc-id.pt

Key words: world wide web, dynamic content management, web proxy, detecting distributed cycles, distributed garbage collection.

Abstract: To fulfill Vannevar Bush's Memex and Ted Nelson's Hyper-Text vision of a world-size interconnected store of knowledge, there are still quite a few rough-edges to solve. There are no large-scale mechanisms to enforce referential integrity in the WWW. The weight of dynamically generated content w.r.t. static content has progressed enormously. Preserving accessibility to this type of content raises new issues. We propose a system, comprised of a distributed web-proxy and cache architecture, to access and automatically manage web content, static and dynamically generated. It is combined with an implementation of a cyclic distributed garbage collection algorithm for wide-area memory. It correctly handles dynamic content, enforces referential integrity on the web, and is complete w.r.t minimizing storage waste.

1 Introduction

To fulfill Vannevar Bush's Memex(Bush, 1945) and Ted Nelson's Hyper-Text(Nelson, 1972; Nelson, 1988) vision of a world-size interconnected store of knowledge, there are still quite a few rough-edges to solve. These must be addressed before the world wide web can be regarded as an effective and reliable world wide knowledge repository. This includes safely preserving static and dynamic content as well as performing storage management in a complete manner. An effective world wide knowledge repository, whatever its implementation, should enforce some fundamental properties: i) allow timely access to content, ii) preserve all referenced content, regardless of how it was created, and iii) completely and efficiently discard everything else.

1.1 Knowledge Repositories

There are no large-scale mechanisms to enforce referential integrity[1] in the WWW; broken links prove this. For some years now, this has been considered a serious problem of the web(Lawrence et al., 2001). This applies to several types and subjects of content, e.g., i) if a user pays for or subscribes some service in the form of web pages, he expects such pages to be reachable all the time, ii) archived web resources, either scientific, legal or historic, that are still referenced, need to be preserved and remain available, and iii) dynamically generated content should also be accounted and it should be possible to preserve different execution results with time information.

Broken links, i.e., the lack of referential integrity of the web, is a dangling-reference problem. With regard to the web this has several implications: annoyance, breach of service, loss of reputation and, most importantly, effective loss of knowledge. When a user is browsing some set of web pages, he requires such pages to be reachable all the time. He/she will be annoyed every time he tries to access a resource pointed to from some page, just to find out that it has simply disappeared.

There are several techniques useful in finding pages with similar content(Lawrence et al., 2001). Nevertheless, if the user has payed for, or subscribed to this service, he will consider this broken-link failure as a breach of the service, contracted or agreed upon. So, this situation, apparently and regularly innocuous in more unformal domains, can undermine content providers reputation and imply severe losses and costs(Ingham et al., 1996).

As serious as this last problem, there is another one related to the effective loss of knowledge. As mentioned in earlier works, broken links on the web can lead to the loss of scientific knowledge(Lawrence et al., 2001). We dare to say that, in the time to come, this problem can affect legal and historical knowledge, as these areas become more represented on the web.

It is known that every single document in these fields

[1]Preservation of every resource, still targeted by references, contained in other resources accessible from some defined root.

is stored in some printed or even digital form in some library. But, if this knowledge is not easily accessible, throughout the web, and its content preserved while it is still referenced (and it will be), it can be considered as effectively lost because it will not be read by most people who are not able, or willing, to search for printed copies.

This is not, as yet, a serious situation but, as web content gets older, it will become an important issue(Lawrence et al., 2001). Nevertheless, solutions that try to preserve every and anything can lead to massive storage waste. Therefore unreachable web content, i.e. garbage, should be reclaimed and its storage space reused.

1.2 Dynamic Content

The weight of dynamically generated content w.r.t to static content has progressed enormously. From a few statically disposed web pages, the WWW has become a living thing with millions of dynamically generated pages, resorting to user context, customization, user class differentiation. Today, the vast majority of web content is dynamically generated, shaping the so-called deep-web, and this has been increasing for quite some time now(O'Neill et al., 2003; Bergman, 2001). This content is frequently perceived by users as more up to date and accurate, therefore having more quality. Since this content is generated on-the-fly, it is potentially different every time the page is accessed.

This may be due to different query parameters, different server and session state, or simply because it changes with time. It is ordinary, nowadays, to have several users hit the same page and get very different results either presentation or content-wise. There is differentiation between registered and unregistered users, paying and non-paying users, per-user customizations of web pages based on sex, age, personal preferences and other factors commonly used by marketing techniques.

It is clear that this type of content cannot be preserved by simply preserving the scripting files that generate it. This is specially relevant with content changing over time. It is produced by scripting pages that, although invoked with the same parameters (identical URL), produce different output, at every invocation, or periodically. Examples of these include stock tickers, citation rankings, ratings of every kind, stocks inventories, so called last-minute news, etc. So, changes in produced output, or in the underlying database(s), should not prevent users from preserving content of interest to them, and keep easy access to it.

In fact, data is only lost[2] when actual data sources (e.g. database records) are deleted. Nevertheless, it could become otherwise unavailable causing effective loss of information, because the data would still reside somewhere but inaccessible, since the exact queries to extract it would not be known. Thus, dynamic content, in itself, must also be preserved while it is still referenced and not just the script/pages that generate it.

Furthermore, other pages pointed by URLs included in every reply of these dynamic pages must be preserved, i.e., content dynamically referenced must be also preserved while it is reachable.

1.3 Shortcomings of Current Solutions

Current approaches to the broken-link problem on the world wide web are not able to preserve referential integrity supporting dynamically generated content and minimize storage waste due to memory leaks in a complete manner. Therefore, the web is not effectively a knowledge repository. Useful content, dynamic and/or static, could be prematurely deleted while useless, unreachable content wastes systems resources throughout the web.

Nowadays, content in the WWW is more interconnected than ever. What initially was comprised of a set of almost completely separated sites with few links among them has become a through highly connected web of affiliated sites, portals, ranging from entertainment to public services. It is not uncommon to notice web sites that besides their in-house generated content, link, many times as part of a subscribed service (payed or not), to content produced and maintained in different sites. These referring sites should have some kind of guarantee, in terms of maintenance, that this referred content will still be available while there are subscribers interested in it, i.e., some kind of referential integrity should be enforced. This has been a field of active study for some years now.

Current solutions to the problem of referential integrity(Andrews et al., 1995; Ingham et al., 1996; Moreau and Gray, 1998) do not deal safely with dynamic content and are not complete, since they are not able to collect distributed cycles of unreachable web data.

Some previous work(Andrews et al., 1995; Creech, 1996; Kappe, 1995), while enforcing referential integrity to the web, impose custom-made (or customized) authoring, visualization or administration schemes. However, for transparency reasons and ease of deployment, it would be preferable to have a system that would enforce referential integrity on the web, to content providers and subscribers, in a mostly transparent manner, i.e., based solely on proxying with minor server and/or client extensions.

Previous approaches(Reich and Rosenthal, 2001) to the broken-link problem, replicate web resources in order to preserve them, in an almost indiscriminate fashion, wasting storage space and preserving content no longer referenced. This stems from the goal that included to provide high availability of web content but not to manage storage space efficiently. Thus, they are not complete w.r.t. storage waste, i.e., they do not reclaim useless data.

Thus, only some of the existing solutions attempt to enforce referential integrity on the web and also reclaim

[2]W.r.t its inclusion in dynamic content, output of future executions of scripting pages.

content which is no longer referenced from any root-set (these root-sets may include bookmarks, subscription lists, etc). These solutions(Ingham et al., 1996; Moreau and Gray, 1998), however, are either unsafe in the presence of dynamically generated content, or they are not complete.

1.4 Proposed Solution

The purpose of this work is to develop a system that:

- enforces referential integrity on the web.
  - preserves, in a flexible way, dynamic web content as seen by users.
  - preserves resources pointed by references included in preserved dynamic content.
- performs complete wasted storage reclamation, i.e., is able to reclaim distributed cycles of useless web content.
- integrates well with the web architecture, i.e., based on web-proxies and web-caching.

These properties must be correctly and efficiently combined.

We propose a solution, based on extending web-proxies, web-server reverse proxies, and a distributed cyclic[3] garbage collection algorithm, that satisfies all these requirements. It enforces referential integrity on the web and minimizes storage waste. Furthermore, this solution scales well in a wide area memory system as is the case of the web, since it uses an hierarchical approach to distributed cycle detection.

For ease of deployment, this solution requires no changes in browsers or servers core application code. It just needs deployment of extended web-proxies that intercept server-generated content, provide it to other proxies or to web servers, and fulfill the distributed garbage collection algorithm. Users are still able to access any other files available on the web.

Thus, our approach makes use of a cyclic distributed garbage collector, combined with web-proxies in order to be easily integrated in the web infrastructure. It intercepts dynamically generated content in order to safely preserve every document/resource referenced by it.

We do not address the issue of fault-tolerance, i.e. it is out of the scope of the paper how the algorithm used behaves in the presence of communication failures and processes crashes. Nevertheless, the algorithm used is safe w.r.t. message loss and duplication.

Therefore, the contribution of this paper is a system architecture that integrates with web-proxy facilities, ensures referential integrity, including dynamic content, on the web, minimizes storage waste in a complete manner, and scales to wide area networks.

The remaining of this paper is organized as follows. In Section 2 we present the proposed architecture. The distributed garbage collection (DGC) algorithm used is briefly described in Section 3. Section 4 highlights some of the most important implementation aspects. Section 5 presents some performance results. The paper ends with some related work and conclusions in Sections 6 and 7, respectively.

[3]Therefore, complete.

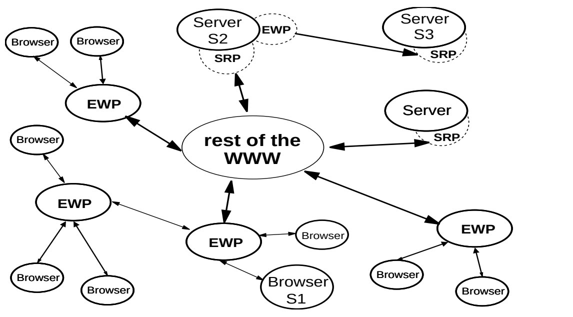

Figure 1: *General architecture of system deployment. Obviously, any number of sites is supported: servers, proxies and browsers.*

2 Architecture

In order not to impose the use of a new, specific, hyper-media system, the architecture proposed is based on regular components used in the WWW or widely accepted extensions to them. The system is designed using a client-server architecture, illustrated in Figure 1, with:

- web servers.
- clients - web browsing applications.
- extended web-proxies (EWPs) - these manage clients requests and mediate access to other proxies.
- server reverse-proxies (SRPs) - intercept server generated content and manage files.

W.r.t the issue of cache coherency at proxies, it is not necessary for the system to function, since it deals with coherency in a conservative manner. However, our system allows refreshing and/or updating of cached content.

The entities manipulated by the system are web resources in general. These come in two flavors: i) html documents that can hold text and references to other web resources, and ii) all other content types (images, sound, video, etc.). Resources of both types can be accessed and are preserved while they are still reachable. Html documents can be either static or dynamically generated/updated. Other web resources, though possibly dynamic as well, are not considered for references to other resources and are viewed, by the system, as leaf-nodes in a web resources graph. Thus, memory is organized as a distributed graph of web resources connected by references (in the case of the web, these are URL links).

We considered, mainly, as cases of web usage:

- i) web browsing without content preservation[4], i.e., standard web usage.
- ii) web browsing with book-marking desired explicitly by the user, either in a page-per-page basis or tran-

[4]this is indicated by the user through book-marking.

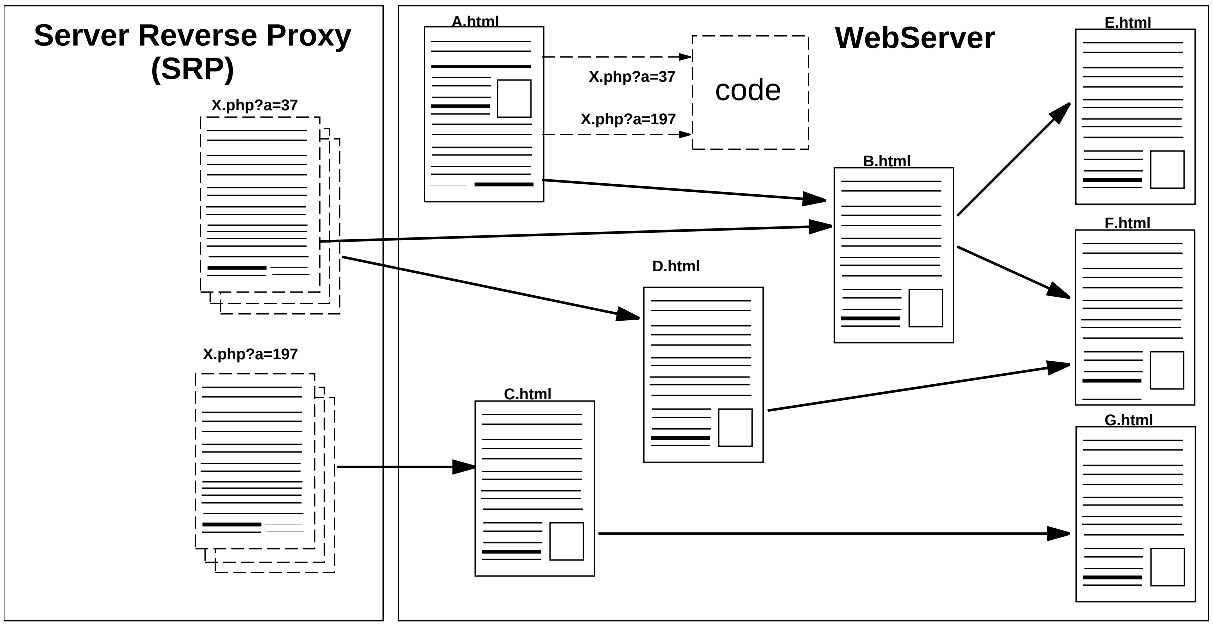

Figure 2: Example web graph with several versions of previously dynamically generated content.

sitively. This enforces referential integrity, preserving all content reachable from the bookmarks set.

From the user point of view, the client side of the system is a normal web browser with an extra toolbar. This toolbar enables book-marking the current page or a URL included in a page as a root page and inform the proxy of such. Nothing prevents running the extended web-proxy in the same machine as the web browser[5].

With this application, a typical user in S1 browses the web, accesses and bookmarks some of the pages from, for example, web-server at site S2 (see Figure 1). Once book-marked, these pages may hold references to other (not book-marked) web resources in site S2. Thus, it is desirable that such resources in site S2 remain available as long as there are references pointing to them. Web resources in other servers (e.g. site S3), targeted by URLs found in content from site S2 are also preserved, while they are still referenced.

The system ensures that web resources in sites remain accessible, as long as they are pointed from some html document. In addition, web resources, which are no longer referenced from any other document, are automatically deleted by the garbage collector. This means that neither broken links nor memory leaks (storage waste) can occur.

Figure 2 presents an example web graph, with dynamically generated web content (the two dynamic URLs) preserved several times, represented like pages over pages. These preserved dynamic pages hold references to different html files, depending on the time (and session information) when they were book-marked. Preserved dynamic content is always stored at the SRP to maintain transparency w.r.t the server.

[5]though it would be obviously more efficient to install a proxy hierarchy.

3 Referential Integrity and Storage Management

To enforce referential integrity and reclaim wasted storage we made use of a cyclic distributed garbage collector for wide area memory(Veiga and Ferreira, 2003) and tailored it to the web.

Briefly, the DGC algorithm is an hybrid of tracing (local collector), reference-listing (distributed collector) and tracing of reduced graphs (cycle detection). Thus, it is able to collect distributed cycles of garbage. Tracing algorithms transverse memory graphs from a root-set of objects and follow, transitively, every reference contained in them, to other objects. Reference-listing algorithm register, for every object, what objects in other sites, are referencing it.

In each proxy there are two GC components: a local tracing collector and a distributed collector. Each site performs its local tracing independently from any other sites. The local tracing can be done by any mark-and-sweep based collector, e.g., a crawling mechanism. The distributed collectors, based on reference-listing, work together by exchanging asynchronous messages. The cycle detectors receive reduced, optimized graph descriptions from a set of other sites and safely detect distributed cycles comprised within them.

GC Structures: The garbage collector components manipulate the following structures to represent references contained in web pages:

- A **stub** describes an outgoing inter-site reference, from a document in the the site to another resource in a target site.
- A **scion** describes an incoming inter-site reference,

from a document in a remote source site to a local resource in the site.

It is important to note that stubs and scions do not impose any indirection on the access to web pages. They are simply DGC specific auxiliary data structures.

GC Roots and Reachability: The root-set of documents for both the local and distributed garbage collectors in each site is comprised of local roots and remote roots: i) local roots are web documents, located in the site and referenced from a special html file[6] managed by the the system; ii) remote roots are all local web documents that are remotely referenced, i.e., protected by scions (see Section 3). These web resources must be preserved even if no longer locally reachable, i.e., reachable from the local root-set.

The root-set of the whole system corresponds to the union of the root-sets in all sites. This way, reachability is defined as the transitive closure of all web documents referenced, either directly or indirectly, from a web document belonging to the root-set just defined. Every other document is considered unreachable and should be reclaimed.

GC Rules: The algorithm obeys to the following safety rules:

- **Clean Before Send:** When a SRP replies to a HTTP request for a page whose content should be preserved, every URL enclosed in it, must be intercepted, i.e., parsed. A corresponding scion to each URL enclosed must be created, if it does not exist already.
- **Clean Before Send:** When a EWP receives a response to a HTTP request for a page whose content should be preserved, every URL enclosed in it, must also be intercepted. A corresponding stub to each URL enclosed must be created, if it does not exist already.

3.1 Local Colletor

The local garbage collector (LGC) is responsible for eventually deleting or archiving unreachable web content. It must be able to crawl the server contents. The roots of this crawling process are defined at each site. They must include scion information provided by the distributed collector (see Section 3.2). Crawling is performed only within the site and lazily, in order to minimize disruption to the web server. The crawler maintains a list of pages to visit. These pages are parsed and references, found within them, to pages in the same server, are added to this list. References found are saved in auxiliary files. These can be re-used later by the crawler, when he re-visits the same page, in another crawl, if the page was not modified.

[6]Named LocalRoot.html, that can be regarded as a bookmarks file, but any html file can be configured to hold the root-set.

Web resources can be created with any authoring tool. Once created in some site, web resources must become reachable in order to be accessible for browsing. This can be done in two ways: i) add a reference to the new resource in the local root-set, or ii) add a reference to the new resource in some existing and reachable document.

If it is necessary to update a page content (static page change or programmatic page update), the page will then be locked and the crawler must wait and will need, for safety reasons, to re-analyze it. This is performed following the links included in both versions (the previous and the new one). Then, after the whole local graph, has been analyzed, the new DGC structures replace (flip) the previous ones. Unreachable web pages can then be archived or deleted.

To prevent race conditions with the garbage collector, newly created resources are never collected if they are more recent than the last collection, i.e., new files always survive at least one collection before they can be reclaimed. Possible floating-garbage[7] in consecutive collections, is minimum, since creation of web resources is a task preformed not that intensively.

Explicit deletion of web resources is extremely error-prone and, therefore, it should not be done (though we provide no means to prevent explicit deletion of a file using the operating system interface). Web resources should only be deleted when they are reclaimed by the garbage collector. This happens when they are no longer reachable either locally or remotely.

3.2 Distributed Colletor

The distributed garbage collector is based on reference-listing(Shapiro et al., 1992) and is responsible for managing inter-site references, i.e., references between local pages to pages placed at other sites (both incoming and out-going). This information is stored in lists of scions and stubs organized, for efficiency reason, by referring/referred site.

From time to time, the distributed collector running on a site sends, to every site it knows about, the list of stubs corresponding to the pages, in the destination site, still referenced from local pages. These lists are sent lazily.

Conversely, the distributed collector receives stubs lists from other sites referencing its pages. Then, it matches stub lists received with corresponding scion lists it holds. Scions without stub counterpart indicate incoming inter-site references that no longer exist. Therefore, the corresponding scion is deleted indicating the page is no longer referenced remotely.

Once a page becomes unreachable both from the site local root and from any other site, it can be deleted by the local collector.

The distributed collector co-operates with the local garbage collector in two ways: i) it provides the LGC a set of pages (target of inter-site references) that must be included in the LGC root-set, and ii) every time the

[7]Resources that are already unreachable but that have not yet been identified by the collector.

LGC completes crawling the site, it updates DGC structures regarding out-going references (stub lists). This update information will be sent later, by the local collector, to the corresponding sites.

3.3 Distributed Cycles

Based on work described in(Richer and Shapiro, 2000), we can estimate the importance of cycles in the web. This research about the memory behavior of the web, revealed that a large proportion of objects are involved in cycles but they amount to a limited, yet not negligible, fraction of storage occupied. We believe that, as the degree of inter-connectivity (its true richness) of the web increases, as well as due to more dynamic content, the number, length, and storage occupied by cycles is also expected to rise.

In the system, special cycle detection processes (CDPs), receive information from participating sites (running EWPs and/or SRPs) and detect cycles fully comprised within them. Several CDPs can be combined into groups. These groups can also be combined into more complex hierarchies. This way, detection of cycles spanning a small number of sites, does not overload higher-level CDPs dealing with larger distributed cycles.

To minimize bandwidth usage and complexity, CDP works on a reduced view of the distributed graphs that is consistent for its purposes. This view may not correspond to a consistent cut, as defined by (Lamport, 1978), but still allows to safely detect distributed cycles of garbage. These distributed GC-consistent-cuts can be obtained without requiring any distributed synchronization among the sites involved(Veiga and Ferreira, 2003).

They are built by carefully joining reduced versions of graphs created in each site. These reduced graphs simply register associations among stubs and scions, i.e., they regard each site as a small set of entry (scion) and exit (stub) objects. This is enough to ensure safety and completeness. Graph reduction is performed, incrementally, in each site.

Once a cycle is detected, the cycle detector instructs the reference-listing algorithm to delete one of its entries (a scion) so that the cycle is eliminated. Then, the reference-listing is capable of reclaiming the remaining garbage objects.

3.4 Integration with the web

The WWW owes, a significant part of its success until now, to the fact that it allows clients and servers to be loosely coupled and different web sites to be administrated autonomously. Therefore, our system does not impose total world-wide acceptance in order to function. Integration with the web can be seen from two perspectives, client and server.

Regular web clients can freely interact with server reverse-proxies, possibly mediated by regular proxies, to retrieve web content. However, they cannot preserve web resources or interfere with the DGC in anyway. Thus, browsing and referencing content will not prevent it from being eventually reclaimed, since these references can be regarded only as weak-references[8]. References contained in indexers are a particular case of these weak-references.

Regular web applications in servers do need not be modified to make use of referential-integrity and DGC services. However, once a file is identified as garbage, the proxy must have some interface with the server machine to actually delete or archive the object. If proxy and server reside on the same machine, this interface can be the actual file system.

Distributed caching is widely used on the web today. It is a cost-effective way to allow more simultaneous accesses to the same web content and preserve content availability in spite of network and server failures. Caching is performed, mainly, at four levels: i) web servers, e.g. dynamically generated and periodically updated content, ii) proxies of large internet service providers, iii) proxies of organizations and local area networks (several of these can be chained), and iv) the very machine running the browser. Due to this structure, the web relies on caching mechanisms that have an inherent hierarchical nature. This can be exploited to improve performance(Chiang et al., 1999).

Hosts performing levels II and III caching are transparent, as far as the system is concerned. They can be implemented in various ways provided they fulfill the HTTP protocol. To perform level I, we propose a solution based on analysis of dynamic content. Server replies are intercepted by the SRPs and URLs contained in them are parsed, before the content is served to requesting clients and proxies. This is not intrusive for applications nor users. Similar techniques have already been applied, as part of marketing-oriented mechanisms (e.g. *bloofusion.com*). These convert dynamic URLs in static ones, to improve site ratings in indexers like *google.com*. They also allow web crawlers to index various results from different executions of the same dynamic page.

4 Implementation

The prototype implementation was developed in Java, mainly for ease of use when compared with C or C++. It simply deploys a stand-in proxy that interprets HTTP-like custom requests to perform DGC operations and relies on a real, full-size web-proxy, running on the same machine, to perform everything else. We intend to address performance issues more closely, thus integrating both sets of requests together in a single application.

Preserving dynamically generated content raises a semantic issue about browser, proxy and server behavior. When a dynamic URL, previously preserved, is accessed, two situations can occur, depending on session information shared with the proxy: i) the content is retrieved as a fresh execution , or ii) the user is allowed to

[8]References that point to a resource but do not keep it from being reclaimed by the garbage collector.

decide, from previously accessed and preserved content, which one he wants to browse.

Dynamic content selection is implemented allowing two configurable default behaviors: i) when a dynamic URL is requested, the browser receives an automatically generated HTML reply, with a list of previously preserved content, provided with date and time information and ii) the very HTML code, implementing the link to the dynamic URL, is replaced with code that implements a selection box, offering the same alternatives as the first option. The first behavior is less computationally demanding on the proxies but the second one is more versatile, in terms of user experience.

Server-side proxies perform URL translations to access corresponding ”invisible” files that hold the actual preserved content. We are currently modifying a widely used open-source web proxy in order to facilitate deployment in several networks.

5 Performance

Global performance, as perceived by users, is just marginally affected. W.r.t URL-replacing mechanisms mentioned before, they are already in practice in several web sites, and users do not perceive any apparent performance degradation. Our system makes use of similar techniques to parse URLs included in dynamic web content. We should stress that, in terms of performance, this is a much lighter operation that URL-replacing.

We determined increased latency in web-servers replies, due to processing in the SRPs. We performed several tests with two widely accessed sets of files, parsing the URLs included in them. These sets were obtained by crawling two international news sites: *bbc.co.uk* and *www.reuters.com* with a depth of four. These sets of files include both static and dynamically generated content.

The *www.reuters.com* test-set comprised 313 files, including 57856 URLs. On average, each file included 184 URLs and took extra 12.7 milliseconds to serve due to parsing. The *bbc.co.uk* test-set comprised 439 files, including 70401 URLs. On average, each file included 160 URLs and took extra 11.8 milliseconds. We intend to gather further performance results, with and without the service, to accurately determine overheads, in load, latency and bandwidth.

6 Related Work

The task of finding broken links can be an automated using several applications(HostPulse, ; LinkAlarm, ; XenuLink,). However, these applications do not enforce referential integrity because, while useful detecting local and remote broken links, cannot prevent them from occurring or reclaim wasted storage. Furthermore, they are not able to handle dynamically generated content in a safe manner. Enforcing referential integrity on the web has been a subject of active study for several years now(Lawrence et al., 2001). There are a few systems that try to correct the broken-link problem and, thus, enforce referential integrity, preserving web content availability.

In (Swaminathan and Raghavan, 2000), dynamic web content is pre-fetched, i.e., cached in advance, based on user behavior identified and predicted using genetic algorithms. Results show that pre-fetching is effective mainly for files smaller than 5000 bytes. Such techniques could be combined with our system in order to handle dynamic content more efficiently while enforcing referential integrity.

LOCKSS(Reich and Rosenthal, 2001) is a open-software based system that makes use of replication, namely spreading, in order to preserve web content. It has been tested in a large environment and it stresses three important requirements: i) future availability of information, ii) quick and easy access to information and iii) reservation of access only to subscribers. There are some fundamental differences to our work: much work has been devoted in LOCKSS to ensure replica consistency, namely using hashing for each document. Storage reclamation is not addressed in LOCKSS since all documents in the system are considered important enough to be preserved forever. Dynamic content is also not addressed. In LOCKSS, the system tries to preserve everything consistent. Ours tries to prevent memory leaks while preserving referential integrity although allowing discrepancy among cached copies of dynamic content.

Author-Oriented Link Management(Creech, 1996) is a system that tries to determine which pages point to a certain one. It describes an informal algebra for representing changes applied to pages, like migration, renaming, deletion, etc. It relies on the usage of custom-made, or customized authoring tools, i.e., referential integrity is not transparently provided to the user or developer. It allows little parallelism and admits manual re-conciliation of data. So, it is more addressed at helping web developers than to preserve referential integrity on a wide scale basis. It does not try to reclaim storage space occupied by useless, i.e., unreachable documents.

Hyper-G(Andrews et al., 1995; Kappe, 1995) is a ”second-generation” hyper-media system that aims to correct web deficiencies and provide a rich set of new services and features. With regard to referential integrity, it is enforced using of a propagation algorithm that is described as scalable. Hyper-G is proposed as an alternative to the WWW. Our system is integrated within and mostly transparent to the current WWW architecture.

The W3Objects(Ingham et al., 1996) approach is also based on the application of a distributed garbage collector to the world wide web. It also imposes a new model extending the WWW based on objects. Therefore it also lacks transparency. It is not complete, i.e., it is not able to reclaim distributed cycles of garbage.

In(Moreau and Gray, 1998) a community of agents is used to maintain link integrity on the web. As in our work, they do not attempt to replace the web but extend it with new behavior. Agents cooperate to provide versioning of documents and maintain links according to a distributed garbage collector. Each site manages tables documenting import and export of documents. It does not address the semantic issues raised when preserving dynamic content.

Thus, existing solutions to referential integrity either do not aim at recycling unreachable documents or are not correct w.r.t. dynamic content or are not integrated with the standard web.

7 Conclusions and Future Work

In this paper we presented a new way of enforcing referential integrity in the WWW, including dynamically generated web content. The fundamental aspects of the system are the following.

- It prevents storage waste and memory leaks, deleting any resources no longer reachable, namely, addressing the collection of distributed cycles of unreachable web content.
- It is safe w.r.t dynamically generated web content.
- It does not require the use of any specific authoring tools.
- it integrates with the hierarchical structure of today's web-proxies and caching.
- it is mostly transparent to user browsers and web servers.
- From the last two properties we can infer that it integrates well with the web. It does not impose its model to the whole web. Sites using this system can still be browsed by regular clients. Conversely, clients using the system can browse any other web sites as well.

Concerning future research directions, we intend to address further the fault-tolerance of our system, i.e., which design decisions must be taken so that it can remain safe, live and complete in spite of process crashes and permanent communication failures.

REFERENCES

Andrews, K., Kappe, F., and Maurer, H. (1995). The hyper-g network information systems. *J.UCS*, 1(4).

Bergman, M. K. (2001). The deep web: Surfacing hidden value. *The Journal of Electronic Publishing*, 7(1).

Bush, V. (1945). As we may think. *The Atlantic Monthly*, (July).

Chiang, C.-Y., Liu, M. T., and Muller, M. E. (1999). Caching neighborhood protocol: a foundation for building dynamic web caching hierarchies with proxy servers. In *International Conference on Parallel Processing*, pages 516–523.

Creech, M. L. (1996). Author-oriented link management. In *Fifth International World Wide Web Conference*, Paris, France.

HostPulse. Broken-link checker, www.hostpulse.com.

Ingham, D., Caughey, S., and Little, M. (1996). Fixing the "Broken-Link'' problem: the W3Objects approach. *Computer Networks and ISDN Systems*, 28(7–11):1255–1268.

Kappe, F. (1995). A Scalable Architecture for Maintaining Referential Integrity in Distributed Information Systems. *Journal of Universal Computer Science*, 1(2):84–104.

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565.

Lawrence, S., Pennock, D. M., Flake, G. W., Krovetz, R., Coetzee, F. M., Glover, E., Nielsen, F. A., Kruger, A., and Giles, C. L. (2001). Persistence of web references in scientific research. *IEEE Computer*, vol 3(2), pp26-31.

LinkAlarm. Linkalarm, http://www.linkalarm.com/.

Moreau, L. and Gray, N. (1998). A community of agents maintaining link integrity in the world wide web. In *Proceedings of the 3rd International Conference on the Practical Applications of Agents and Multi-Agent Systems (PAAM-98)*, London, UK.

Nelson, T. H. (1972). As we will think. In *On-line 72 Conference*.

Nelson, T. H. (1988). Managing immense storage. *Byte Magazine*, 13(1):224–238.

O'Neill, E. T., Lavoie, B. F., and Bennett, R. (2003). Trends in the evolution of the public web 1998 - 2002. *D-Lib Magazine*, 9(4).

Reich, V. and Rosenthal, D. (2001). Lockss: A permanent web publishing and access system. *D-Lib Magazine*, 7.

Richer, N. and Shapiro, M. (2000). The memory behavior of the WWW, or the WWW considered as a persistent store. In *POS 2000*, pages 161–176.

Shapiro, M., Dickman, P., and Plainfoss, D. (1992). Robust, dist. references and acyclic garbage collection. In *Symp. on Principles of Dist. Computing*, pages 135–146, Vancouver (Canada). ACM.

Swaminathan, N. and Raghavan, S. (2000). Intelligent prefetch in www using client behavior characterization. In *8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 13–19.

Veiga, L. and Ferreira, P. (2003). Complete distributed garbage collection, an experience with rotor. In *IEE Proceeding Software (Research Journals) - special issue on Rotor Workshop*, London.

XenuLink. Linksleuth http://home.snafu.de/tilman/.