STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVIII**, Number 3, 2013

A SERVER-SIDE SUPPORT LAYER FOR CLIENT PERSPECTIVE TRANSPARENT WEB CONTENT MIGRATION

DARIUS BUFNEA[(1)] AND DIANA HALIŢĂ[(1)]

Abstract. The migration process of a website's content within a Content Management System almost always implies changes in the site structure as seen by search engines and web clients. This variation leads to some disadvantages, such as misdirecting search engines visitors to old, unavailable, URLs. Even if, over time, search engines adapt to changes in the site's structure, the problem remains unsolved for visitors landing from 3rd party referrers. This paper presents a server-side support layer for client perspective transparent web content migration, layer that automatically maps the old visible structure of a website to the new one implied by the migration process. Some of the advantages of such a mechanism are: reducing incoming dead links from 3rd party referrers, assisting search engines for properly redirecting users or page rank and SERP conservation.

1. Introduction

In today's Internet, more and more web sites are being built using a Content Management System (CMS). From the top one million websites, as classified by Alexa.com, 22.5% of them are built on a CMS platform and WordPress occupy a percentage of 12.4% of the total[1]. And these numbers continue to grow. More and more websites are being powered by Content Management Systems, among their advantages, we can highlight the following:

Received by the editors: April 16, 2013.

2010 *Mathematics Subject Classification.* 68U35 - Information systems, 68M11 - Internet topics.

1998 *CR Categories and Descriptors.* H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *Web-based interaction*; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia – *Navigation* .

Key words and phrases. CMS migration, web content migration, transparent 404 not found redirection.

This paper has been presented at the International Conference KEPT2013: Knowledge Engineering Principles and Techniques, organized by Babeş-Bolyai University, Cluj-Napoca, July 5-7 2013.

- Easy maintenance: content's maintainer must not have knowledge of design / HTML, he does not need to be a web programmer to handle website's content;
- Consistency of design is preserved: the content from all authors is presented with the same, consistent design;
- Easy migration from one presentation to another, from one design to another: site's graphic can be changed without need of rebuilding the website;
- Regular security updates;
- Full control over the elements related to search engine optimization. Content freshness is a factor that helps a lot because search engines prefer a site with a content updated on a daily or at least weekly basis;
- Multiple roles for users with different levels of rights;
- Additional functionality offered: 3rd party plugins;
- Web based administration: editing anywhere, anytime removes bottlenecks;
- Sites are self-organized based on categories, pages, post links. Navigation is automatically generated adjusted: site's menus are typically generated automatically based on the database content and links will not point to non-existing page.

All above advantages are the main reasons that more and more websites are being built based on a Content Management System. More over, based on the same favors, old sites built on static technologies are being migrated within a CMS, considering also the scalability of such a system.

There are also situations, when a CMS based web site must be migrated to a different, new CMS. In many cases, the current CMS can not support new goals or does not have all the functionality or features needed to achieve them. Migrating from one CMS to another is necessary from several points of view:

- the old CMS might be considered deprecated;
- there is the possibility of choosing an open source CMS granting easier access to support;
- CMS scalability;
- highlighting the newest web technologies: CSS3, Ajax, HTML5.

2. Migration process challenges

Migration of a web site to a new CMS can be done either manually or automatically. In the case of a static technologies based web site, the migration process will involve most likely manual operations. On the contrary, migration between two popular CMS will imply some automatically performed actions.

The migration process it's not only a problem related exclusively to content migration. One of the most difficult part of migrating to a new CMS is mapping the old content to a new one, in addition to mapping the old logical organization model to the new one as required by the new CMS.

The complexity of the migration process can have a gradual solution, following three steps: dividing the content into categories, estimating needed time for migration and migration reevaluation based on guidance.

The first step of the migration process is about classifying different types of content, depending on the outcome of its analysis and derived types - these types shall mean all categories in which the content is subdivided. This creates two types of rules that must be identified: those rules which relate specifically to particular content, needed also after migration, and rules relating to what can and what can not be automatically migrated. These rules are useful in defining the priorities and the efforts which must be made. At this point, a decision can be taken about what can be automatically migrated and what can't. Generally it is desired to automate as much of the content that must be migrated.

The second step is about guiding and estimating migration needed time, i.e. compare automatic migration with manual migration required time.

The final step involves evaluating the automatic migration process. The biggest problem which arises when choosing automatic migration is content's structure and its regularity. On the other hand, manual migration to another CMS requires in some cases viewing, editing and manually moving the content, which probably will lead to a waste of resources, time and effort.

How difficult is migration of a website's content within a CMS or from one CMS to another?

- Competition between CMS providers translates into a difficult transition between them.
- Using the same CMS for many years makes migration difficult; a consequence could be the ability to run an inconsistent code.
- Consolidation content management platforms in a single system can be logistically a nightmare.
- Content that is meant to be migrated is not limited to text / html, but may include graphic content resources (images), multimedia content, which does not appear in the database structure (that graphic content resides as a file in a file system).
- In some cases, content migration is hindered by its organization. For example, a page in PHP Website is organized as sections reunion, including an order relation which is timeless. Such multisection page

turns naturally into a series of temporal posts (lacking a time attribute).
- Once the content migration is complete, it is necessary to change internal link's meaning, so that it would point to the appropriate page from the new site structure.
- If the site is very dynamic, i.e. the information on it are updated regularly, what happens when the site is in a state of transition? Time spent on updates will be doubled, the information must be renewed also in the old site and the new one (which has not yet been released).
- The new site structure will not be instantly visible in Internet by search engines; however search engines have the ability to adapt (sooner or later) to this new structure. On the other hand, third party referrers that link to the migrated website probably will never adapt to the new site structure, this operation implying manual intervention.

The migration process implies not only moving the content within the new CMS, the migration, either automatically or manually performed, should also take into consideration the transparent behavior of the web site from visitor perspective, either a web browser or a search engine crawler. One common problem induced by such a migration is exposing to the web a certain content to a new, different URL as it was presented in the old site. After migration, this will be reflected in lots of misleading visitor coming from search engines or third party referrers. Besides not serving the proper content to visitors, the newly migrated website can lose page rank or other in-time gather benefits induced by back links or social shares. This paper presents a server-side support layer for client perspective transparent web content migration, layer that automatically maps the old visible structure of a website to the new one implied by the migration process. Some of the advantages of such a mechanism are: reducing incoming dead links from 3rd party referrers, assisting search engines for properly redirecting users or page rank and SERP conservation.

Similar solutions as the one presented in this paper to the above problems have been proposed and implemented, mostly as plugins, inside all major CMS today ([5], [6], [7]). However, these solutions, in order to properly redirect a user or a crawler in case of a 404 not found request, only take into consideration some in-time gather data based on user behavior, such as target or exit pages in users' session or users' chosen pages from a custom search that follows the 404 response. Our server-side support layer is trying to match new URLs to old ones based on content similarity, or other semantic related information such as: URLs, the query given to the search engines or referrer's content similarity with the linked content. Also, such a solution has the advantage

of being implementable prior to the release of the new web site, having no dependence on in-time gather statistical data based on user behavior.

3. Types of content migration

The following formal notations are general enough for any CMS (or static content website) to any CMS migration, but we'll highlight them on our real life scenario CMSes implied in the migration process: phpWebsite as the source CMS and Wordpress as the target CMS.

The content which appears both in the old site and in the new site, is presented in two different ways. We are dealing either with static content (i.e. a file that exists in the file system of the web server, usually an external resource such as a .pdf or .jpg file) or with dynamic content, taken from a CMS' database.

A) Static content migration. When we migrate such a content, even if its URL is changing, the base name of the file remains the same.

Example:

```
oldsiteURL: http://oldsite/oldpath/filename
newsiteURL: http://oldsite/newpath/filename
```

We define it as perfect match, $similarity($`oldsiteURL, newsiteURL`$) = 1$.

B) Dynamic content migration. We describe the content which is found either at an old or new URL as consisting of two parts. The first part is represented by the content corresponding to the page template (i.e. the master page), and the second part is the absolute content which is usually stored in the database. Actually, the absolute content is the one being migrated. When comparing the similarity of old and new content, we'll take into discussion only the absolute content, in order to avoid the noise induced in the similarity algorithm by the master pages' HTML code.

Example:

For Wordpress the absolute content can be extracted either from the database or directly from its associated URL. Within a Wordpress' master page, the absolute content can be retrieved as the inner HTML of the div having the `content` id.

We'll designate by OP and by NP a page from the old site and a page from the new site, respectively. In Wordpress, absolute content is represented by:

- pages $NP_{\neg T}$ (timeless articles, i.e. pages)
- posts NP_T (temporal articles, i.e. posts)
- reunion of posts (i.e. a category)

- subset of posts in a category

In phpWebsite, absolute content is represented by:

- single section page: OP_{SS}. Such a page is usually migrated in a $NP_{\neg T}$.
- multiple section page: $OP_{MS} = \bigcup content(section_i)$. Such a page is migrated either in a single $NP_{\neg T}$, if its sections are time independent, or each of its sections becomes a separate post NP_T tagged in the same category. The migration of the temporal posts in Wordpress has encountered difficulties because at least in phpWebsite, its data model stores no time related information, in relation to the date of creation or modification of a page section.

When we compare the similarity of a content from the old site which corresponds to a page with multiple sections (page which in terms of presentation is shown as a category of posts in the new site) it is useful to compare the similarity of the absolute content of the old site content with the absolute content (in the new site) of a subset of posts in that category. The substantiation for this claim is presented in section 4 of our paper.

C) A static content of the old site, especially an HTML file, is migrated as dynamic content within the new CMS. This is useful for integrating file's content in the new look and feel of site. In this situation, the similarity function should be computed based on the absolute content from the new site and on the file's content from the old site.

A possible approach when we have to migrate a site between two CMSes would be to compare the absolute content within the databases maintained by the two CMSes. We have rather prefer to determine the absolute content or the custom content involved in the similarity comparison based on its web presentation because:

- This method is more general, in comparison with the one which effectively compares content from databases. It can also be taken into consideration when migrating a static web site.
- We would not cover cases listed above.
- There are no additional privileges required, for examples access rights to the CMS database. Querying content via its web presentation URL is much easier since no knowledge is required about the organization model or the database structure of the CMS.
- Matching process should be done from the web client perspective.

4. Algorithm and implementation

This section of our paper presents the algorithm (and its implementation details) used for matching old site URLs to new ones. In order to match these URLs, our algorithm makes use of a similarity function, but the method

is not dependent of a certain similarity function. Future work may imply comparison of the results obtained with different similarity functions, from their speed and matching accuracy point of view. Our approach is general enough to be applied to any similarity function, or to allow the replacement of this function with minimal effort.

Information processing is not done in real time (i.e. while the user gets a 404 not found response). Rather than trying a real time computation, we choose instead a batch processing approach: a previously run program implements a similarity algorithm in order to identify URLs with similar content. There are several reasons to do this. First of all, in both sites there are thousands of valid URLs. This would imply that the complexity of the algorithm to be at least equal to the cardinal of the cartesian product of the two sets: old site's and new site's URLs. Running a real time computation will induce delays in serving a response to the web client. Secondly, the batch program runs completely independent of the CMS core, its language of implementation does not necessarily depend on an API exported by the CMS.

For a quick match we use the cosine similarity algorithm [3]. This algorithm has a fast and well-implemented implementation within Apache Lucene [2]. Apache Lucene is a high-performance, full-featured text search engine library that provides a fast and tested implementation for at least the similarity algorithm. It is an open source project available for free download. Moreover, its API allows users to modify the function used in the similarity algorithm by implementing matching providers, in order to get better or faster results when comparing two absolute contents.

The formal presentation of the algorithm implemented by the batch process follows:

```
For each oldsiteURL having static content do
   Identify the newsiteURL which points to the same static)
      content (based on base filename) (section 3, A)
   If this newsiteURL was identified then
      eliminate the oldsiteURL from the URLs list which must be processed
   EndIf
EndFor
For all unprocessed oldsiteURLs do
   If it is a static content URL then
      absolute_content = the effective content (section 3, C)
   else
      absolute_content = content(oldsiteURL) - content(page template)
         (section 3, B)
   EndIf
   Identify the newsiteURL so the absolute_content has the best
```

```
      similarity with the oldsiteURL's absolute_content
   The matching pair is inserted in a table, together with the current
      date, the similarity and the best similarity ever
EndFor
```

One might wonder why we stored in a database the current similarity and also the best similarity ever. A category evolves in time, because new announcements might appear in it. In such a case, the similarity obtained when comparing the category presentation URLs is decreasing. We will have maximum similarity if we make the comparison with a subset of older announcements (i.e. the newest announcements from the old site).

Example: In Student News category, semantically speaking, a visitor might want to see the newest articles and he may want to be redirected to the newest ones, not to a snapshot page that match perfectly (from the content point of view) the news in the way they appear in the old site.

In order to speed up the running time, a threshold experimentally determined can be used. If the similarity of two compared URLs is greater than a threshold, both old and new URLs can be removed from their sets and not get used anymore in the comparison process.

The deployment of the results map in a real live situation such as a production web site, was done directly in the Apache configuration file (i.e. the appropriate .htaccess or *.conf file) using the `Redirect permanent` directive [4]. This method guarantees the independence of the CMS and its technology. Another advantage is the fact that this file can be deployed by the user for most web hosting service providers.

More over, this Apache configuration file can be overwritten after future runs of the algorithm and it can be easily integrated in the `web client -> web server -> web application -> backend database server` workflow, practically without any intervention to the new CMS configuration, core, plugins or content.

5. Results and evaluation

In order to demonstrate the benefits of our approach we have performed some experiments related to the number of not found (i.e. 404 HTTP response code) pages requested by users landing on our site from an external referrer. By external referrer we understand either a search engine or a 3rd party referrer that links our site. These experiments were run for 30 days, both on the old site before migration and on the new site as well.

The number of not found requests addressed to our test site before migration was relatively small, around 1.9% of the request from an external referrer

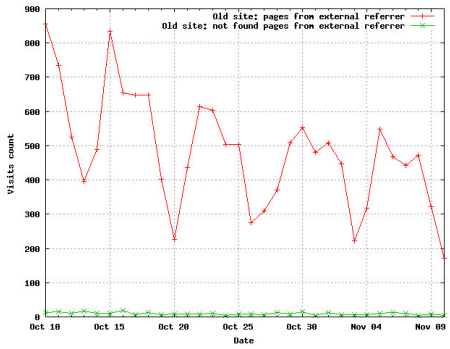

FIGURE 1. Old site: number of pages accessed from external referrer and number of not found pages accessed from external referrer

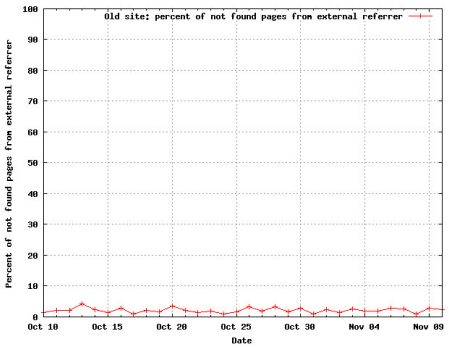

FIGURE 2. Old site: percent of not found pages accessed from external referrer

generating a 404 response (Fig. 2). These responses where mainly generated for requests coming from a 3rd party referrer.

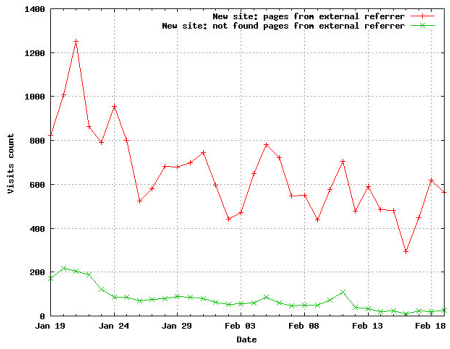

FIGURE 3. New site: number of pages accessed from external referrer and number of not found pages accessed from external referrer

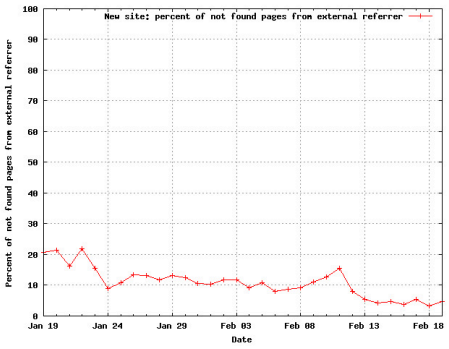

FIGURE 4. New site: percent of not found pages accessed from external referrer

After migration, the number of not found requests made via an external referrer increased naturally to an average of 11.2% for 30 days (Fig. 4). Search engines are quickly adapting to modifications in a site's structure, about one week after migration they were correctly redirecting the traffic to proper landing pages (Fig. 5, Fig. 7).

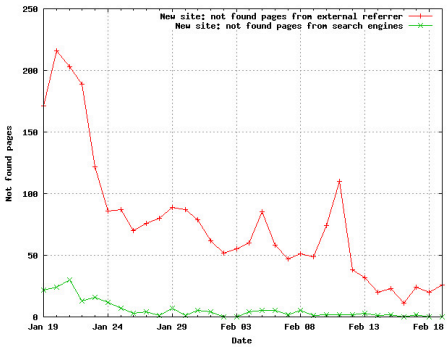

FIGURE 5. New site: adaptation of the search engines to the new structure of the site

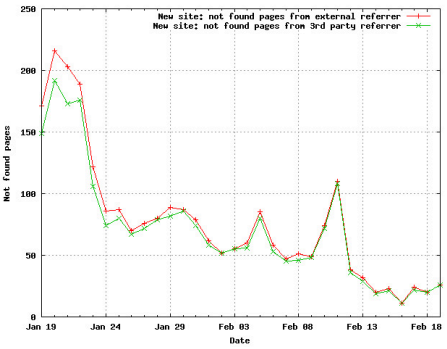

FIGURE 6. New site: Most of not found pages are coming from 3rd party referrer

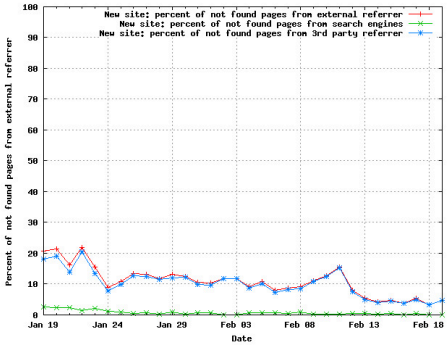

FIGURE 7. New site: percent of not found pages, not found pages from search engines, not found pages from 3rd party referrer

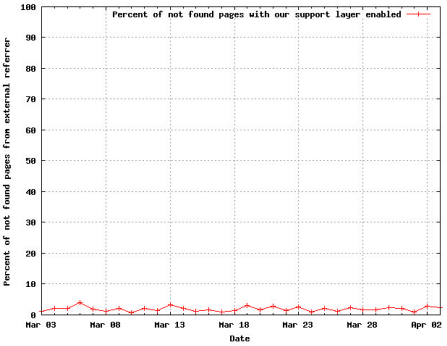

FIGURE 8. Percent of not found pages having an external referrer with our support layer enabled

After migration, most of not found request were made via a 3rd party referrer (6).

Figure 7 depicts the percent of not found requests made via external referrers in a 30 days time interval, after the migration process had been completed.

In the first week after migration, around 1.9% of the requests made via a search engine (from the total number of requests made via an external referrer) were generating a 404 response code, this value dropping to 0.23% in week four. In the matter of requests made via a 3rd party referrer, in the first week after migration we measured that 14.7% percent of these requests were for not found

pages, this value dropping to 4.25% in week four, still above the average of not found responses that where generated before migration.

Figure 8 shows the percent of not found response after we embedded our support layer inside the newly migrated version of our test site. The 404 not found responses have dropped to an average of 1.6% counted for a 30 days time interval, bellow the average of the old site. In the batch run of the similarity algorithm we have identified pairs of URLs having the cosine similarity higher than 0.7 for around 93% of the migrated content. This threshold was experimentally chosen, in fact the similarity distribution in the [0, 1] interval was either above 0.8 for matching URLs (i.e. correctly identified pairs of URL), or bellow 0.3 for arbitrary pairs.

6. Conclusions and future work

In this paper we have advanced a server-side support layer for client perspective transparent web content migration within a CMS. We have pointed out the need of such a layer and the challenges in implementing it. Also, we have proposed a method, covering both the theoretical and practical aspects, for implementing this layer by mapping URLs from the old site to the ones in new site based on their content similarity.

We are currently focused in evaluating different similarity functions in order to improve the matching process and its speed. An improvement which can be brought into discussion refers to giving different weights in the similarity algorithm to the various properties of the content such as: URL, page headings, page title, key words (when the user is coming from a search engine). Another option to be taken into consideration is redirecting the user to a similar page, from the content point of view, as the page where he is coming from (i.e. the page within our site most similar with the referrer). The main goal remains the same: succeeding in user's redirection, no matter where he comes from, even if he comes from a third party referrer.

References

[1] Leena Rao, *WordPress Now Powers 22 Percent Of New Active Websites In The U.S.*, August, 2011, TechCrunch

[2] Apache Lucene, http://lucene.apache.org/, *A high-performance, full-featured text search engine library*

[3] Amit Singhal, *Modern Information Retrieval: A Brief Overview*, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2011, 24 (4): 35-43

[4] mod_alias - Apache HTTP Server, http://httpd.apache.org/docs/current/mod/mod_alias.html

[5] Zyxware Technologies, *Search404: Automatically search for content when a 404 error occurs*, http://drupal.org/project/search404

[6] *Dynamic404 - Logical error-pages*, http://www.yireo.com/software/joomla-extensions/dynamic404

[7] *404 Redirected Wordpress plugin*, http://wordpress.org/extend/plugins/404-redirected/

(1) Babeş-Bolyai University, Department of Computer Science, Cluj-Napoca, Romania

E-mail address: bufny@cs.ubbcluj.ro

E-mail address: diana.halita@ubbcluj.ro