International Journal of Advanced Science and Technology
Vol.83 (2015), pp.1-12
http://dx.doi.org/10.14257/ijast.2015.83.01

# Design of an Efficient Migrating Crawler based on Sitemaps

Deepika[1] and Dr Ashutosh Dixit[2]

[1,2]*Computer Engineering Department, YMCAUST, Faridabad, India*
[1]*deepikapunj@gmail.com,* [2]*dixit_ashutosh@rediffmail.com*

***Abstract***

*As the size of web keep on growing, the job of a web crawler has become more cumbersome in covering maximum web. There are enormous numbers of web pages available over the web but out of them a smaller number is available to user. A sitemap is an XML file in which a list of URLs of that particular site is mentioned. The sitemap protocol may play a very important role in covering maximum web. In this paper, the information provided by sitemap protocol is used for the purpose of crawling the quality web pages. With the help of sitemap, web crawlers will maintain their repository up-to-date. It also tries to make user to access maximum pages over the web by covering as maximum as possible while crawling. It also helps crawler to visit the pages based on their change frequency and downloads updated pages only, thereby reduces unnecessary network traffic.*

***Keywords****: web crawler, sitemap, freshness, coverage, Migrating agent*

## 1. Introduction

Search Engines are the most common and widely used medium of finding information on the web. User enters a keyword for searching the information and on the basis of that keyword search engines searches their databases and give the results related to users' query. These databases are created from a repository maintained by web crawler. Web crawler crawl the web, downloads the documents and stored them in search engines repository. They continuously crawls the web to get more relevant and new information. So, web crawler is an important module of any search engines. There are many issues [6] related to design an efficient web crawler. User interest [14] can also be considered while designing a crawler but in this paper, more emphasis is on maximum coverage and freshness of database while keeping the network traffic low. As the size of web grows exponentially, it is very difficult to crawl to the whole web and maintained the freshness of search engines repository. Even with presence of massive resources, crawlers are not able to do their task efficiently.

Sitemaps may help web crawler to discover all the links present on a particular web page. It is an XML file that lists all the links of a web page and also the other information about that web page *e.g.*, when the page was last modified, how frequently the web page will change and how much the importance of any link in comparison to another links present on that web page. Sitemaps also help in extracting structure of a web page and then this extracted structure can be used for many purposes [16, 17].

Although without sitemap crawler may discover most of the links but with sitemap it will do the task more efficiently. Following are the reasons for this:

1 Large size of Web Site: - it may be possible that due to the large size of website, some links may be missed by a crawler while downloading.

2. New Web Site: - it may be the case that web crawler always follows the same pattern of crawling and due to this it misses the new entry.

ISSN: 2005-4238 IJAST

3. Less External Links: - some websites has less links to other websites and crawler crawls to the web by following one page to other. So having less number of links causes crawler to rarely visit that particular site.

From above mentioned reasons, it may be justified that with the help of sitemap crawler may work more efficiently while downloading the web pages.

Here an example of Gmail is taken where sitemap of Gmail technical support is designed with the help of website www.web-site-map.com. Sitemap protocol for Gmail technical support is given below:

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xsi:schemaLocation="http://www.sitemaps.org/schemas/sitemap/0.9
http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
<url>
        <loc>http://www.gmailtechnicalsupport.com/</loc>
        <changefreq>daily</changefreq>
        <priority>1.00</priority>
</url>
<url>
        <loc>http://www.gmailtechnicalsupport.com/services</loc>
        <changefreq>daily</changefreq>
        <priority>0.85</priority>
</url>
<url>
        <loc>http://www.gmailtechnicalsupport.com/privacy-policy</loc>
        <changefreq>daily</changefreq>
        <priority>0.85</priority>
</url>
<url>
        <loc>http://www.gmailtechnicalsupport.com/contacts</loc>
        <changefreq>daily</changefreq>
        <priority>0.85</priority>
</url>
<!-- Generated by www.web-site-map.com -->
</urlset>
```

The sitemap may consist of some essential fields and some optional fields as discussed below: -
Loc- it is the place where URL is specified of particular web page

Lastmod: it is an optional field, which specifies when the page was last updated

Changefreq: it is an optional field, which specifies the frequency of web page changed like always, hourly, daily, weekly, monthly, never.

Priority: it is also an optional field, which specifies the priority in comparison with other URLs present in the page.

Other then above mentioned parameters various others parameters can be added to sitemap *e.g.*, information about images, sitemap for mobile, *etc*. For large websites whose sitemaps are very large in size are difficult to download. So instead of downloading large size sitemap, SitemapIndex files are used. This allows to big sitemaps breaking into smaller sitemaps and keeps their entries in SitemapIndex file. The size of these SitemapIndex file is small and they are easy to download and manage.

 

## 2. Related Work

Brandman [2] introduces the concept of servers providing some metadata to the crawlers such as last modification and file size. Their main purpose was to make web servers friendly with the crawlers and make the databases rich and fresh.

Damien Lefortier Yandex *et al.*, [15] worked on a different section of web pages called as ephemeral new pages. These are those pages on which users' interest grows within hours as they appear but remain only for few days. He found the sources of such pages and then re-visit them in order to get newly created such pages at faster rate.

Giovanni [3] and Richard [4] use the sitemap protocol for semantic web browsers. They worked on RDF datasets. They extended the sitemaps protocol for large semantic datasets. They proposed other Meta information with site maps so that the crawling performance may be improved. Instead of crawling large website, it will download only dump of it that is stored in sitemap. This dump will also help in crawling missed or scattered links of that website by downloading only sitemap. They were trying to provide novel functionalities to both servers and clients. By deploying semantic sitemap, they showed effective consumption of resources for saving large quantity of semantically structured data.

Uri [5] *et al.*, uses sitemaps protocol in crawling algorithm. They suggested with the help of sitemap, duplicates pages can be removed. It also helps in finding missing links. By getting all information about all the links present in particular webpage, it helps in crawling maximum coverage. With the help of sitemaps they showed difference in classic crawling and sitemap crawling.

Gurpreet Singh [8] *et al.*, discussed various SEO techniques, process and categories. They discussed Microsoft SEO case study. Microsoft SEO tool kit has three components. Sitemap is one of the SEO tool and others are robot inclusion and site analysis. They also discussed the location of sitemap i.e. where it has to be placed. According to them, it should be placed in robot.txt file.

S S Vishwakarma *et al.*, [13] proposed a modify approach for crawling. It uses last visit time of crawler and apply filter at server side. This filter will check this last visit time and return list of only Urls that are updated after crawler last visit. This is a query based approach. HTTP uses it GET method to know the list of updated Urls list. Through this approach now crawler is revisiting only updated pages.

Najork *et al.*, [11] approach suggests that breadth-first search is a good crawling strategy, as it tends to discover high-quality pages early on in the crawl. On early it means that as crawling increases progressively the quality of web pages deteriorates. It uses connectivity based metric Page Rank to measure quality of a page. It also increases the overall download rate and reduces the server from overloading.

## 3. Design of an Efficient Migrating Crawler based on Sitemaps

In this paper, with the help of sitemaps crawler tries to crawl web more efficiently such that updated information always get stored into database, while keeping the network load low. The sitemaps are used to provide checks at various levels before downloading the page. These checks are done by web crawler. With the help of the some of the fields which is used are given below:

*Changefrequency*: It contains the frequency values at which a web page is changing e.g. hourly, weekly, monthly or yearly. This field may help crawler to categorise the webpages on the basis of their change frequency.

*Lastmod*: It contains the time of a web page when it was last updated. This field may help crawler to download only updated or modified pages to their databases.

Now with the help of these fields, a migrating crawler is designed. It will distribute its migrating agents over the web and tries to get maximum coverage. It will apply checks at various levels before revisiting and downloading the page.

### 3.1 Architecture of Proposed Work

A migrating Crawler is a crawler which has its agents called as Migrating Agents. On behalf of crawler, these agents do crawling over the web. Crawler sends its agent to different server and on the server side, agents do the processing *e.g.*, downloading, compressing etc at server side only and return the results back to the crawler side.

The proposed work has following subsystems:

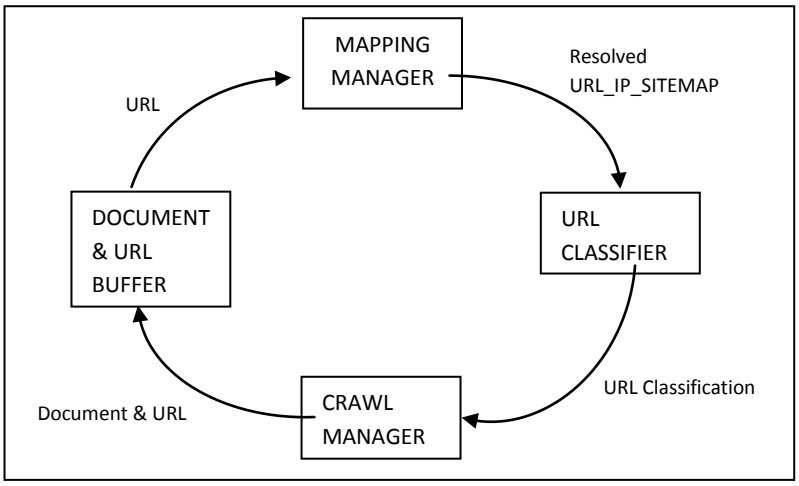


**Figure 1. Workflow of Proposed Architecture**

1. Mapping Manager
   It maps URLs into their IP addresses and also stores sitemap of corresponding URL with the help of Sitemap Generator in resolved URL-IP-Sitemap Queue.
2. URL Classifier
   It will classify the URLs on basis of their changed frequency like hourly, daily, weekly, monthly etc. And then inform the crawl manager about their classification.
3. Crawl Manager
   It will send the migrants to their assigned URL server as provided by URL Classifier. After calculating their revisit, documents are downloaded.
4. Document & URL Buffer
   This is a buffer for storing documents send by different migrating agents from different servers. From this buffer URLs are extracted and send back to Mapping Manager for further processing.

Here in proposed architecture, with the help of sitemap, crawling is trying to improve by migrating agents. Migrating agents will work in different way and crawls the web efficiently. Various modules are involved in working of Migrating Agents. The working of each module is described in next section.

The general working of Proposed Migrating Crawler is as follows:- URL_IP_sitemap values which are generated by Mapping Manager are used by URL Classifier which will classify URLs on the basis of their change frequency. It will change daily, hourly, weekly, monthly or never. Now agents are assigned to each such classified list and visit the URLs according their change frequency. On reaching the websites, whether to download the pages or not will depend on whether it has updated or obsolete copy. If agent is visiting

 

first time, then agent doesn't have any last crawled time of the webpage and it simply downloads the page. But if agent is revisiting it then it will check the last crawled time of webpage with the last mod values of webpage which it has from sitemap. If last crawled value is newer then last mod values of websites then it will not download the page as agent already has its updated page with it. But if value of last crawled is older then it shows there has some modifications has been done and agent has old copy with it. So it will download the page and replace its copy with the new updated one.

The general architecture of proposed work is as follows:

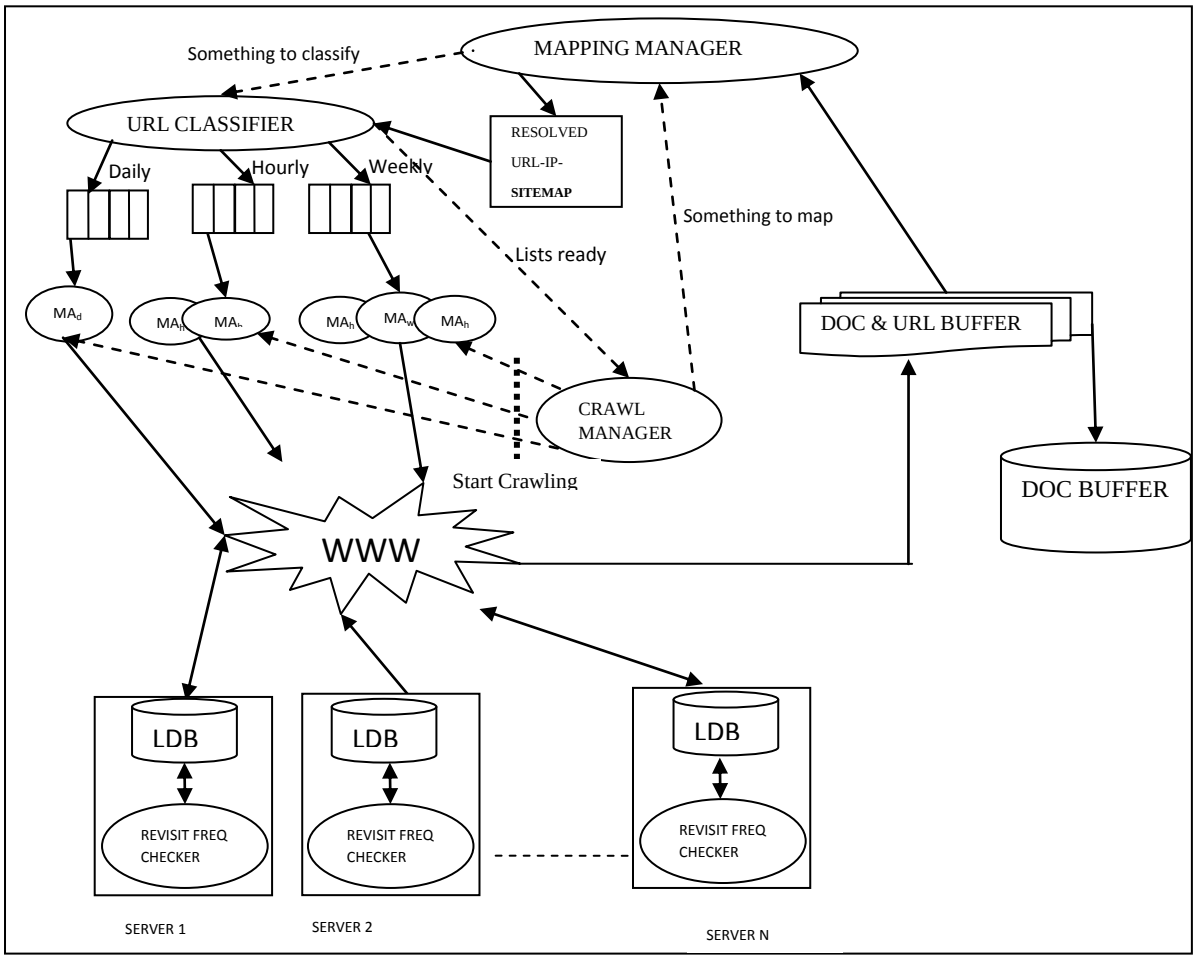


**Figure 2. Proposed Architecture of Migrating Crawler**

The general working algorithm works as:

```
Step1:URL_List_Sitemap= Mapping Manager ()
Do Forever
    2:Pick URLs from URL_List_Sitemap;
    3:Li=URL_classifier(s(i), URL_List);
        //Li is the list of URL of i category
        Where i= daily, weekly, monthly
    4:Signal(agents_ready);
    5:Doc_buffer= Crawl_manager(Li);
    6:(URL,Doc)= Extract(doc_buffer);
    7:Take URLs and give it to URL_classifier;
```

**Figure 3. Algorithm for General Work Flow**

**3.1.1 Mapping Manager:** Mapping manager [10] will provide resolved URL_IP pair. It will get IP for corresponding URL from DNS resolver and stored the pair in a Queue. In addition to this pair with the help of sitemap generator [1], sitemap of every URL is also provided and stored with the same resolved URL_IP pair. After filling data in queue it will send the signal to URL classifier to start their work.

```
Mapping Manager ()
Step1: Wait (Something to map)
    2: While (URL-IP Queue is not empty)
    3: Take a URL-IP pair from the Queue;
    4: If the IP is blank
        4.1 Call DNS resolver to resolve URL for IP;
        4.2 Store the Resolved URL in the Resolved URL Queue;
    5:  Call SiteMap Generator to create sitemap of every resolved
        URL;
    6:  Store the sitemap with each URL_IP pair;
    7: Signal (something to classify);
```

**Figure 4. Algorithm for Mapping Manager**

The Mapping Manager has used following data structures:

**3.1.1.1 URL-IP Queue:** It consists of a queue of unique seed URL-IP pairs. The IP part may or may not be blank. It acts as an input to the mapping manager.

**3.1.1.2 Resolved URL-IP-Sitemap Buffer:** It stores resolved URLs and also their corresponding sitemap. Sitemaps are generated with the help of Sitemap Generator. It acts as input to the URL Classifier.

**3.1.2 URL Classifier:** After getting signal from Mapping Manager, it will pick the URL and classify them on the basis of their change frequency which it will get from sitemap.

 

```
URL_classifier(S(i))
Do forever
Step1: Wait (Something to classify);
       2: Check S(i) for frequency change;
       3: If (changefreq==daily)
            3.1:  Add to Ld;
       4: If (changefreq==weekly)
            4.1: Add to Lw;
       5: If (changefreq==monthly)
            5.1: Add to Lm;
       6: Signal (list_ready);
```

**Figure 5. Algorithm for URL Classifier**

According to its change frequency whether it is daily, weekly, monthly *etc*., changes, lists are maintained and corresponding URLs are added to them.

After maintaining the lists, it will send signal to crawl manager to inform that lists are ready for migrating agents to crawling.

**3.1.3 Crawl Manager:** It is the responsibility of crawl manager to create multiple migrating agents. It will name them according to the list of URLs provided by URL classifier. After getting signal from crawl manager, it will assign the migrating agents to each list supplied by URL classifier. Then it will send signal to migrating agents to start crawling to the WWW.

```
Crawl_manager(Li)
Do forever
Step1: create multiple migrating agents;
       2: Wait (list_ready);
       3: Assign URLs List to corresponding agent;
       4: Submit to Mobile_agent(WWW);
       5: Signal (start_cralwing);
        6: Signal (something to map);
```

**Figure 6. Algorithm for Crawl Manager**

**3.1.4 Mobile Agent:** Mobile agents are waiting for signal from their crawl manager to start their work. They get their list of URLs and now they start downloading the data corresponding to each URL. On reaching the server, if it will download the data second time then before downloading it will check from its database whether it has this data updated copy or not. If it has old one only then it will download that data otherwise it will not.

```
Mobile_agent(WWW)
Step1: Wait (start_crawling);
Do forever
      2:  agents visit to remote server of assigned URLs;
      3: Before downloading check its last mod value;
      4: If lastmod_time(URL) is newer than last_crawled_time(URL)
            4.1 Download page;
         Else
      5: Ignore(URL);
      6: Store the downloaded page and URLs into doc_buffer;
      7: Signal (something to map);
```

**Figure 7. Algorithm for Mobile Agent**

The Mobile agents during their working use the following data structures:

**3.1.4.1 Local Buffer:** It is a buffer used by the migrants for storing the downloaded documents locally. Before downloading the documents, revisit frequency checker whether download is required or not. It basically check whether the document to be download is modified or updated with the existing copy of that document.

**3.1.4.2 Document and URL Buffer:** This buffer is used to store the recently downloaded documents sent by migrants. From this buffer only URLs are sent back to the URL Classifier for classification.

## 4. Performance Analysis

The migrating Crawler, designed in this work, has been implemented on Java platform. The performance was compared with a conventional method of crawling without sitemap. Various tests have been performed on different websites to observe the difference in crawling with and without sitemap. Here, it is assumed that pages are of fixed size.

The Test1 was conducted by using YMCA website. Website was crawled by both conventional and proposed new method of crawling. The results obtained are discussed in next section. For simplicity few links are shown here to verify the results.

Results obtained for **www.ymcaust.ac.in/computers** are shown in Tables 1a), 1b) & 1c)

**Table 1a). First Crawling Results (Both Crawler)**

| ANCHOR TEXT | URL | CONTENT RECEIVED(bytes) |
|---|---|---|
| Chairman message | http://www.ymcaust.ac.in/computers/index.php/chairman-s-message | 13450 |
| Faculty | http://www.ymcaust.ac.in/computers/index.php/faculty | 10823 |
| Labs | http://www.ymcaust.ac.in/computers/index.php/labs | 14549 |
| Courses | http://www.ymcaust.ac.in/computers/index.php/courses | 15883 |
| B.Tech Syllabus | http://www.ymcaust.ac.in/computers/index.php/b-tech-syllabus | 96216 |
| M.Tech Syllabus | http://www.ymcaust.ac.in/computers/index.php/m-tech-syllabus | 11762 |
| Updated Time Table | http://www.ymcaust.ac.in/computers/index.php/updated-time-table | 11450 |
| Notices | http://www.ymcaust.ac.in/computers/index.php/notices | 11928 |

On revisit proposed crawler to same link following changes has been observed:

**Table 1b). Revisit Crawling Results (Proposed Crawler)**

| ANCHOR TEXT | URL | CONTENT RECEIVED(bytes) |
|---|---|---|
| Updated Time Table | http://www.ymcaust.ac.in/computers/index.php/updated-time-table | 11450 |
| Notices | http://www.ymcaust.ac.in/computers/index.php/notices | 11728 |

 

Whereas Conventional crawling to same link has following results:

**Table 1c). Revisit Crawling Results (Conventional Crawler)**

| ANCHOR TEXT | URL | CONTENT RECEIVED |
|---|---|---|
| Chairman message | http://www.ymcaust.ac.in/computers/index.php/chairman-s-message | 13450 |
| Faculty | http://www.ymcaust.ac.in/computers/index.php/faculty | 10823 |
| Labs | http://www.ymcaust.ac.in/computers/index.php/labs | 14549 |
| Courses | http://www.ymcaust.ac.in/computers/index.php/courses | 15883 |
| B.Tech Syllabus | http://www.ymcaust.ac.in/computers/index.php/b-tech-syllabus | 96216 |
| M.Tech Syllabus | http://www.ymcaust.ac.in/computers/index.php/m-tech-syllabus | 11762 |
| Updated Time Table | http://www.ymcaust.ac.in/computers/index.php/updated-time-table | 11450 |
| Notices | http://www.ymcaust.ac.in/computers/index.php/notices | 11928 |

From the above shown results of both the types of crawling *i.e.*, by proposed crawling and by conventional crawling, it has been observed that on revisit former crawling only visit changed or modified links whereas later crawling visit all links and thus wasted network resources.

Summarized results of full website by Proposed Crawler are shown below:

**Table 1d). Crawling Results (Proposed Crawler)**

| Parameters | Conventional Crawling | New Crawling with Sitemap |
|---|---|---|
| Number of Links | 141 | 148 |
| Total Data Downloaded(Bytes) | 1401300 | 1501233 |
| Crawl Time(sec) | 273.44 | 185.45 |
| Total Pages Downloaded | 141 | 148 |
| Total Pages Downloaded on revisit | 145 | 40 |
| Total Data Downloaded on revisit | 1402387 | 397531 |
| New & Updated Data Downloaded on revisit | 1087 | Nil |

The Test2 was conducted on **www.ngfcet.in/computer_science_engineering**

**Table 2. Crawling Results (Proposed Crawler)**

| Parameters | Conventional Crawling | New Crawling with Sitemap |
|---|---|---|
| Number of Links | 474 | 503 |
| Total Data Downloaded(Bytes) | 10240104 | 10910000 |
| Crawl Time(sec) | 937.729 | 250 |
| Total Pages Downloaded | 474 | 503 |
| Total Pages Downloaded on revisit | 476 | 154 |
| Total Data Downloaded on revisit | 10283311 | 3036580 |
| New & Updated Data Downloaded on revisit | 43207 | Nil |

The Test3 was conducted on **www.titsbhiwani.ac.in/departments/department-of-computer-engineering**

**Table 3. Crawling Results (Proposed Crawler)**

| Parameters | Conventional Crawling | New Crawling with Sitemap |
|---|---|---|
| Number of Links | 174 | 223 |
| Total Data Downloaded(Bytes) | 10347381 | 2730000 |
| Crawl Time(sec) | 613.522 | 210 |
| Total Pages Downloaded | 174 | 223 |
| Total Pages Downloaded on revisit | 180 | 25 |
| Total Data Downloaded on revisit | 10704187 | 306053 |
| New & Updated Data Downloaded on revisit | 356806 | Nil |

The conventional method of crawling crawls to limited number of links and it also downloads all documents whether they have changed or not and thus, wasted network resources. On the contrary, this proposed new crawling method crawls to the web with the help of sitemap. Now it will cover utmost URLs and downloads only that documents that have changed and thus utilize the network resources in efficient manner.

## 5. Efficiency of Sitemap

There are many benefits of sitemap for both web users and web crawlers'. Following are the advantages while using sitemap in crawling process:

### 5.1. Web Coverage

By comparing the coverage area by Conventional Crawling and Proposed Crawling, it is observed that later one will have better area of covering the web as it will visit more number of links.

TEST 1:

Increase in %Coverage (D) = 148-141/141 ≈5%

TEST 2:

Increase in %Coverage (D) = 503-474/474 ≈6%

TEST 3:

Increase in %Coverage (D) = 223-174/174 ≈28%

So, this coverage will increases as the size of website increases. In this test size of websites are small.

### 5.2 Preserves Bandwidth

It also preserves bandwidth by reducing the network traffic by downloading only modified pages on revisit of crawler. This can be seen from above experiments where total number of links on first and second visit is same for conventional crawling whereas less for new crawling method. In this simulation, it has been assumed that pages are of fixed size and therefore, links corresponding to each page has taken into the consideration for looking at the consumption of bandwidth by conventional crawler.

TEST 1:

Total number of links on revisit by Conventional Crawler= 148

Total number of links on revisit by Proposed Crawler= 40

%age consumption of bandwidth by conventional crawler= 148-40/148=72%

TEST 2:

 

%age consumption of bandwidth by conventional crawler = 503-154/503=69.38%
TEST 3:
%age consumption of bandwidth by conventional crawler = 223-25/223=88.7%

Thus, by getting information about last modified date of web page from sitemap, new crawling method now downloads only that pages that are modified or updated. This will help in saving bandwidth and also reduces network traffic.

### 5.3 Co-operation between Migrating Agents

Now with the help of sitemap data, migrating agents can be scheduled. By scheduling, agents will crawl the web in more efficient manner. Database will now have rich in information as compared to databases that are crawled by normal agents.

So, by including sitemap in crawling will give better results as compared to one without sitemap. Crawling will be done in efficient and better way. Below shown the comparison of crawler performance with and without sitemap.

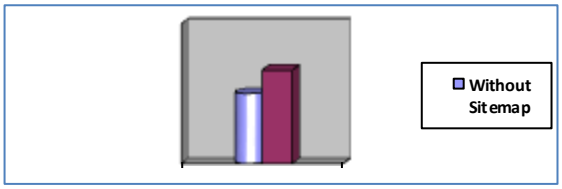


**Figure 8. Performance Evaluation**

With the help of sitemap, performance of crawling can be enhancing. It will get fresh and up-to-date database.

## 6. Conclusion

In this paper, several fields of sitemaps are used to improve the crawling process. These fields have some values which are used at different steps of crawling. Various checks are applied at different points in order to improve the performance of crawler. By simulating this proposed approach, it is observed that now crawling with the help of sitemap gave us better results in terms of coverage, up-to-date data and uses less network resources. It covered all the links and also in less time as compared to crawling without sitemap. Results also shown that on revisit only changed or modified links were crawled instead of all links. This will maintain the freshness of repository in efficient manner.

## References

[1] http://www.sitemaps.org/protocol.php
[2] O. Brandman, J. Cho, H. Garcia-Molina, N. Shivakumar, "Crawler-friendly web servers", Workshop on Performance and Architecture of Web Servers (PAWS), **(2000)** June.
[3] G. Tummarello, "A sitemap extension to enable efficient interaction with large Quantity of linked data", Presented at W3C Workshop on RDF Access to Relational Databases, **(2007)**.
[4] R. Cyganiak, H. Stenzhorn, R. Delbru and S. Decker, "Semantic sitemaps: Efficient and flexible access to datasets on the semantic web", The Semantic Web: Research and Applications, **(2008)**.
[5] U. Schonfeld and N. Shivakumar, "Sitemaps: above and beyond the crawl of duty",-Proceedings of the 18th international conference, **(2009)**.
[6] D. Ashutosh Deepika, "Web Crawler Design Issues: A Review", published in International Journals of Multidisciplinary research Academy (IJMRA), **(2012)** August.
[7] "Microsoft SEO Toolkit," http://www.iis.net/learn/extensions/iis-search-engine-optimization-toolkit/managing-robotstxt-and-sitemap-files.
[8] G. Singh Bedi and Ms. A. Singh, "Analysis of Search Engine Optimization (SEO) Techniques", published in International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, **(2014)** March.

 

[9] Creating google sitemaps files.http://www.google.com/support/webmasters/bin/ topic.py?topic=8467.
[10] A. K. Sharma, J. P. Gupta and D. P. Agarwal, "PARCAHYD: A Parallel Crawler based on Augmented Hyper text Documents", communicated to IASTED International Journal of computer applications, **(2005)** May.
[11] M. Najork and J. L. Wiener, "Breadth-First Search Crawling Yields High-Quality Pages", WWW'01, presented in 10th International World Wide Conference, **(2001)**, pp. 114-118.
[12] S. Mishra, A. Jain and Sachan, "A Query based Approach to Reduce the Web Crawler Traffic using HTTP Get Request and Dynamic Web Page", published in International Journal of Computer Applications (0975-8887), vol. 14, no. 3, **(2011)**, pp. 8-14.
[13] S. S. Vishwakarma, A. Jain and A. K. Sachan, "A Novel Web Crawler Algorithm on Query based Approach with Increases Efficiency", published in International Journal of Computer Applications (0975 – 8887), vol. 46, no. 1, **(2012)** May.
[14] Deepika and A. Dixit, "Capturing User Browsing Behaviour Indicators", published in, Electrical & Computer Engineering: An International Journal (ECIJ), DOI : 10.14810/ecij.2015.4203, vol. 4, no. 2, **(2015)** June, pp. 23-30.
[15] D. Lefortier Yandex, L. Ostroumova Yandex and E. Samosvat Yandex, "Timely crawling of high-quality ephemeral new content", arXiv:1307.6080v2 [cs.IR], **(2013)** July 24.
[16] Deepika and A. Dixit, "Document structure based Filtering in Migrating Crawler- A Review", presented in International Conference Paradigms shift in Management and Technology at Faridabad, **(2015)** March 9-10.
[17] Deepika and A. Dixit "A Novel Approach for Document Structure based Filtering in Migrating Crawler", published in International Journal of YMCA UST Faridabad, **(2014)**.

## Authors

**Deepika Punj**, is working as assistant professor in Department of computer engineering at YMCA University of Science and Technology, Faridabad, India. She is having total 10 years of experience in teaching. Currently, she is doing research in the area of Internet technologies. She has authored papers in national and international journals

**Ashutosh Dixit**, received his PhD and M. Tech. in Computer Engineering from MD University Rohtak, India in the years 2010 and 2004 respectively. He is presently serving as Associate Professor in the department of computer engineering at YMCA University of Science & Technology, Faridabad India. He has published around 70 research papers in various International journals and conferences. His research interests include Internet Technologies, Data Structures and Mobile and Wireless networks.

 