International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-2 Issue-3, February 2013

Implementing Mobile Crawler Using JINI Technology to Reduce Network Traffic and Bandwidth Consumption

R.G. Tambe, M.B. Vaidya

Abstract*— To search any information on the web users extensively use the search engines. As the growth of the World Wide Web exceeded all expectations, the search engines rely on web crawlers to maintain the index of billions of pages for efficient searching. The web crawlers have to interact with millions of hosts and retrieve the pages continuously to keep the index up-to-date. According to literature survey, most of the network traffic and bandwidth is consumed by web crawlers so instead of using them, we are proposing mobile crawler developed using JINI technology with the help of remote page selection, filtration and compression at web servers and not search engine.

***Index Terms*— JINI Technology, Mobile Agent, Mobile Crawler, web Crawler**

I. INTRODUCTION

The WWW (World Wide Web) is a global database where the information is in the form of textual data, images, video, and audio and so on. To extract the user relevant information, through users query in the search engine, search engine depends on mobile crawlers [10] to extract the information. A mobile crawler searches the web servers and downloads all the web pages that are relevant as well as irrelevant to the user with respect to the keywords in the users query. So users do not get relevant information.

In order to download a document, the crawler picks up its seed URL [8], and depending on the host protocol, downloads the document from the web server. The search engines rely on massive collections of web pages that are acquired with the help of web crawlers, which traverse the web by following hyperlinks and storing downloaded pages in a large database that is later indexed for efficient execution of user queries.

A crawler for a large search engine [9] has to address two issues. First, it has to have a good crawling strategy, i.e., a strategy for deciding which pages to download next. Second, it needs to have a highly optimized system architecture that can download a large number of pages per second while being robust against crashes, manageable, and considerate of resources and web servers. Search engines [13] maintain comprehensive indices of documents available on the web to provide powerful search facilities.

Web crawlers are used to recursively traverse and download web pages (Giving GET and POST commands) for search engines to create and maintain the web indices. The needs of maintaining the up to date pages in the collection cause a crawler to revisit the web sites again and again.

A web crawler [1] consumes a significant amount of network bandwidth and other resources by accessing the web resources at a fast speed. This affects the performance of the web server considerably. A significant amount of resources of underlying network are consumed to build a comprehensive full text index of the web.

II. LITERATURE SURVEY

Search engines have become important tools for Web navigation. In order to provide powerful search facilities, search engines maintain comprehensive indices of documents available on the Web. The creation and maintenance of Web indices [6] is done by Web crawlers, which recursively traverse and download Web pages on behalf of search engines.

Analysis of the collected information is performed after the data has been downloaded. In this research, we propose an alternative, more efficient approach to building Web indices based on mobile crawlers [1,6]. Our proposed crawlers are transferred to the source(s) where the data resides in order to filter out any unwanted data locally before transferring it
back to the search engine.

A web crawler is a relatively simple automated program, or script, that methodically scans or ”crawls” through Internet pages to create an index of the data it’s looking for;
these programs are usually made to be used only once, but they can be programmed for long-term usage as well. There are several uses for the program, perhaps the most popular being search engines using it to provide webs surfers with relevant web sites.

In traditional web crawling technique web crawler visits the web server in order to download the pages, downloaded pages are then send back to the search engine where they are selected and filtered out according to the users query and are then indexed by the search engine. Due to these reason network traffic increases and bandwidth consumed by web crawler is in great extend.

Issues with existing crawling technique:

1. The mobile crawlers that always stay in the memory of the remote system occupy a considerable portion of it. This problem will further increase, when there are a number of mobile crawlers from different search engines. All these mobile crawlers will stay in the memory of the remote system and will consume lot of memory that could have otherwise been used for some other useful purposes.

Manuscript published on 28 February 2013.
* Correspondence Author (s)
R.G. Tambe*, Department of Computer Engineering, Amrutvahini College of Engineering, Sangamner, India.
Prof. M.B. Vaidya, Department of Computer Engineering, Amrutvahini College of Engineering, Sangamner, India.

2. It can also happen that the remote system may not allow the mobile crawlers to reside permanently in its memory due to security reasons.
3. In case a page changes very quickly then the mobile crawler immediately accesses the changed page and sends it to the search engine to maintain up-to-date index. This will result in wastage of network bandwidth and CPU cycles etc.

III. PROPOSED MOBILE CRAWLER APPROACH

The studies of [4,5,9], proposed distributed and parallel crawling systems to increase the coverage and to decrease the bandwidth consumption but these systems just distribute and localized the load but does not help much in reducing the load. The studies of [6,7], proposed web crawling approach based on mobile crawlers powered by mobile agents. These mobile crawlers can exploit the information about the pages being crawled in order to reduce the amount of data that needs to be transmitted to the search engine.

These mobile crawlers move to the resources that need to be accessed. After accessing a resource, mobile crawlers move on to the next server or to their home machine, carrying the crawling results in their memory. The main advantage of mobile crawling is localized data access, remote page selection, filtering and compression.

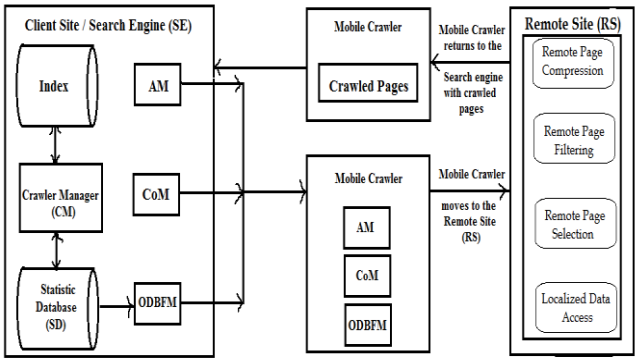


FIGURE 1: Proposed Mobility Based Crawling System

As shown in the figure 1 mobile crawler developed using JINI technology is transmitted from search engine to the web server along with the users query, downloads the pages locally at the web server.

Main Components of the proposed system:

1. Localized Data Access.
2. Remote Page Selection.
3. Remote Page Filtration.
4. Remote Page Compression.

Localized Data Access:

In the context of traditional search engines a stationary crawler [1,10] is an HTTP (Hypertext Transfer Protocol) client which tries to recursively download all documents managed by one or more Web servers. Due to the HTTP request/response paradigm, downloading the contents from a Web server involves significant overhead due to request messages, which have to be sent separately for each Web page to be retrieved. Using a mobile crawler, we reduce the HTTP overhead by transferring the crawler to the source of the data. The crawler can then issue all HTTP requests locally with respect to the HTTP server. This approach still requires one HTTP request per document but there is no need to transmit these requests over the network anymore.

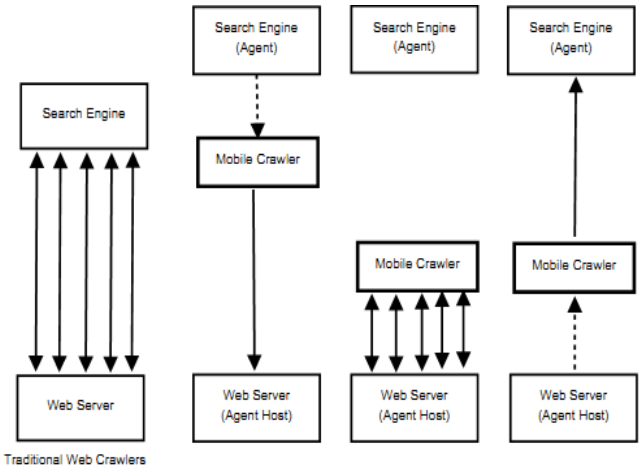


FIGURE 2: Localized Data Access

Following formulas have been describing the network load *L* as a function of the number of crawled pages *N*. *Si* denotes the size of an object (e.g., message, Web page) in KB.

$L_{traditional}(N) = N \times (Si_{request} + Si_{response} + Si_{content})$…(1)

$L_{mobile}(N) = N \times (Si_{content}) + 2 \times Si_{crawler}$…(2)

The formula for the network load caused by a mobile crawler assumes that the mobile crawler is transmitted back to its crawler home along with the retrieved pages. Based on these formulas it derives another function describing the savings *S* in network load due to mobile crawlers.

$S(N) = L_{traditional}(N) - L_{mobile}(N)$

$= N \times (Si_{request} + Si_{response}) - 2 \times Si_{crawler}$ …(3)

By using averages for the size of the crawler, the size of HTTP request and response messages, it can derive the savings as a linear function in the number of pages.

$Si_{request} = 0.1KB$…(4)

$Si_{response} = 0.1\ KB$…(5)

$Si_{crawler} = 1.5KB$…(6)

$S(N) = 0.2KB \times N - 3KB$…(7)

Remote Page Selection:

Once a mobile crawler has been transferred to a Web server, it can analyze each Web page before transmitting it to the search engine. This allows mobile crawlers to determine whether the page is relevant with respect to a particular application domain. Web pages considered relevant are stored within the crawler and are eventually transmitted over the network when the mobile crawler returns to its home.

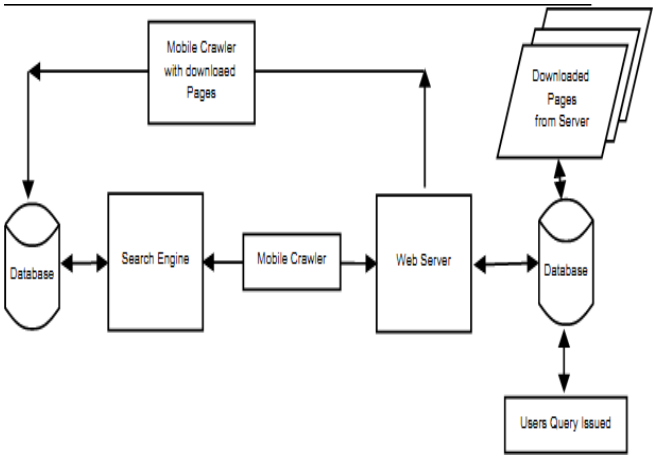


FIGURE 3: Remote Page Selection

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The formulas given below focus on the remote page selection feature and neglect the saving in network load L due to localized data access as analyzed in the previous section. The percentage of pages considered relevant by the mobile crawlers is expressed as an additional factor SF in the formulas.

$L_{traditional}(N) = N \times Si_{content} \ldots (8)$

$L_{mobile}(N, SF) = (SF \times N) \times Si_{content} + 2 \times Si_{crawler} \ldots (9)$

The selection factor *SF* in the second formula relates to *N* since it scales the number of pages transmitted over the network. Again, it is interested how soon mobile crawlers start to outperform traditional crawlers and derive the savings function *S()*. This time, *S()* is a function in *N* and SF.

$S(N, SF) = L_{traditional}(N) - L_{mobile}(N, SF)$

$= (1 - SF) \times N\, Si_{content} - 2 \times Si_{crawler} \ldots (10)$

Assuming the same average numbers for crawler and page size, it can derive the following linear savings function *S()*.

$Si_{crawler} = 1.5\ KB \ldots (11)$

$Si_{content} = 6\ KB \ldots (12)$

$S(N, SF) = 6\ KB \times N \times (1\text{-}SF) - 3\ KB \ldots (13)$

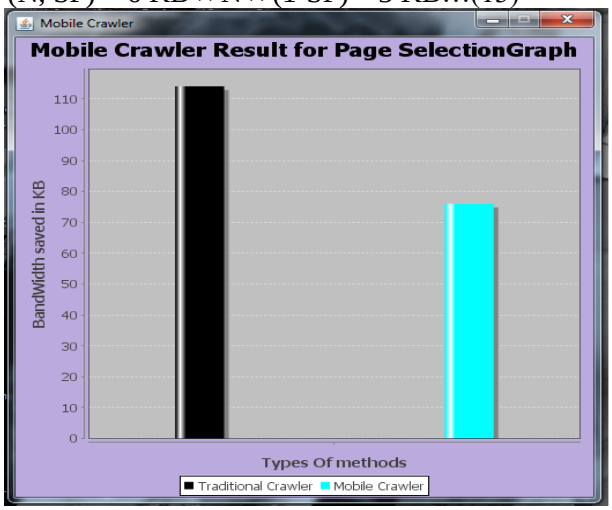


FIGURE 4: Remote Page Selection

Remote Page Filtration:

Remote page filtering [1,3,5,10] extends the concept of remote page selection to the contents of a Web page. The idea behind remote page filtering is to allow the crawler to control the granularity of the data it retrieves. With stationary crawlers, the granularity of retrieved data is the Web page itself. This is because HTTP only allows page-level access. For this reason, stationary crawlers always have to retrieve the entire page before the indexer can extract the relevant page portion. Depending on the ratio of relevant to irrelevant information, significant portions of network bandwidth are wasted by transmitting useless data.

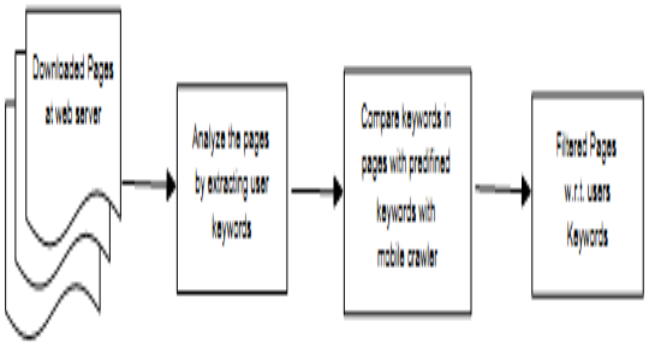


FIGURE 5: Remote Page Filtration

A mobile crawler addresses this problem through its ability to operate directly at the data source. After retrieving a page, a mobile crawler can filter out all irrelevant contents keeping only information, which is relevant to the search engine for which the crawler is working. To get an idea about how significant the potential savings are, it has derived the formulas that describe the situation. First, it established the network load functions *L()* for the traditional and the mobile crawling approach. For simplicity, the overhead due to HTTP request and response messages is neglected. As before, *Si* denotes object size.

$L_{traditional}(N) = N \times Si_{content} \ldots (14)$

$L_{mobile}(N, FF) = N \times (FF \times Si_{content}) + 2 \times Si_{crawler} \ldots (15)$

As in the last section, the load function for the mobile crawling approach is a function in the number of crawled pages *N* and a factor *FF* (filter factor). Here, factor *FF* states the percentage of page content used to represent the page. To estimate the potential savings due to mobile crawling it derive the savings function *S()*.

$S(N, FF) = L_{traditional}(N) - L_{mobile}(N, FF)$

$= (1 - FF) \times N \times Si_{content} - 2 \times Si_{crawler} \ldots (16)$

This savings function is very similar to the result of the last section. This is due to the fact that page filtering is represented in the formulas the same way as page selection. The difference is that page filtering scales the size of the page and not the number of pages (of course, this does not make a difference from the mathematical point of view). It set the parenthesis in the formulas to emphasis this using the values for page and crawler size as before, it get the following linear savings function.

$Si_{crawler} = 1.5\ KB \ldots (17)$

$Si_{content} = 6\ KB \ldots (18)$

$S(N, SF) = N \times ((1\text{-}FF) \times 6\ KB) - 3\ KB \ldots (19)$

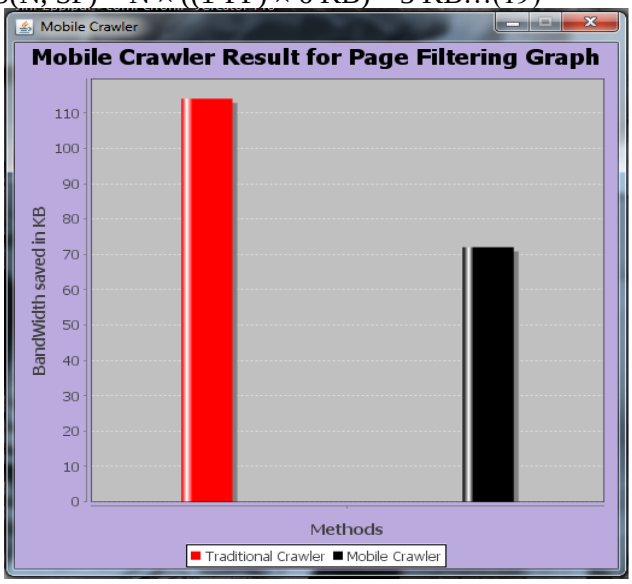


FIGURE 6: Remote Page Filtration

Remote Page Compression:

Remote page selection and filtering are two important characteristics that directly reduce the network traffic caused by Web crawlers. Both perform well in the context of specialized search engines, which cover a certain portion of the Web only. However, the situation is different for a crawler [1,2,4] which is supposed to establish a comprehensive full text index of the Web.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

In this case, as many Web pages as possible need to be retrieved by the crawler. Techniques like remote page selection and filtering are not applicable in such cases since every page is considered relevant.

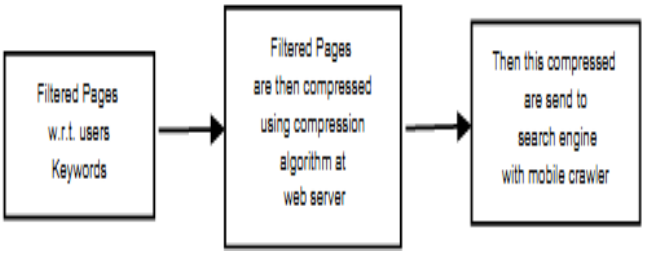


FIGURE 7: Remote Page Compression

As before, it estimate the benefits by deriving load functions *L()* describing the network load caused by traditional and mobile crawlers. This time it focuses on the effects of page compression and neglects the other features.

$$L_{traditional}(N) = N \times S_{content} \ldots(20)$$

$$L_{mobile}(N, CR) = CR \times (N \times S_{content} + 2 \times Si_{crawler}) \ldots(21)$$

The load function for mobile crawlers is a function in the number of crawled pages *N* and the achieved compression ratio *CR*. The load function states that the compression ratio relates to both, the page content and the crawler code. This is due to the fact that our prototype system compresses not only the crawled pages but also the crawler code itself. To estimate the benefits of remote page compression it derive the savings function *S()*.

$$S(N, CR) = L_{traditional}(N) - L_{mobile}(N, CR) = (1- CR) \times N \times Si_{content} - 2 \times CR \times Si_{crawler} \ldots(22)$$

Using our average numbers for page and crawler size, it get the following linear representation of the savings function.

$$Si_{crawler} = 1.5\ KB \ldots(23)$$

$$Si_{content} = 6\ KB \ldots(24)$$

$$S(N, CR) = N \times (1 - CR) \times 6\ KB - CR \times 3KB \ldots(25)$$

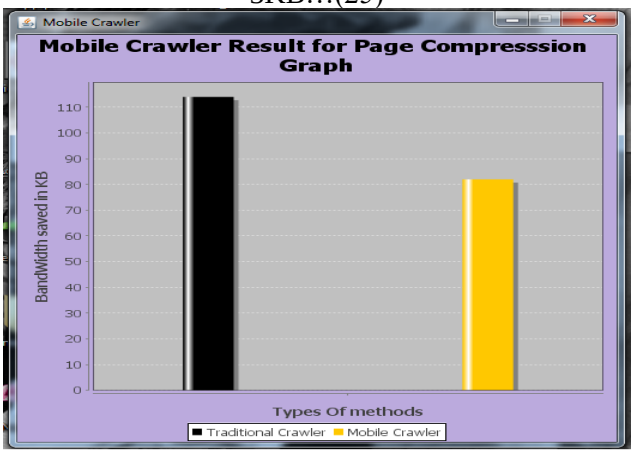


FIGURE 8: Remote Page Compression

IV. COMBINED EFFECT

So far, it is examined the benefits of mobile crawling in an isolated fashion neglecting the fact the above parameters must be examined in combination in order to get a more realistic picture of the performance of mobile crawling. For example, remote page compression can always be activated and will further reduce the network load independent of any other features. The set of features, which can be used by a mobile crawler to reduce the total network load, depends on the type of search engine for which the crawler is working. For mobile crawlers two characteristic features of search engines are important: The first is whether the search engine tries to establish a comprehensive Web index. The second is whether the search engine relies on a full text index or a compact index.

Given these combinations of applicable features it derive general load functions *L()* based on the individual load functions identified in sections 3.1 through 3.4. The general load function *L()* will serve as the baseline for an analytical comparison of mobile and traditional crawling techniques.

$$L_{traditional}(N) = N \times (Si_{request} + Si_{response} + Si_{content}) \ldots(26)$$

$$L_{mobile}(N, CR, SF, FF) = CR \times ((SF \times N) \times (FF \times Si_{content}) + 2 \times Si_{crawler}) \ldots(27)$$

Deriving our savings function *S()* from these load functions leads to the following equation:

$$S(N, CR, SF, FF) = L_{traditional}(N) - L_{mobile}(N, CR, SF, FF) = N \times (Si_{request} + Si_{response} + (1- CR \times SF \times FF) \times Si_{content}) - 2 \times CR \times Si_{crawler} \ldots(28)$$

Although this function looks complicated, it can limit our analysis to the portion containing the filter, selection and compression factors which scale the size of the page content to be transmitted. By neglecting the less significant terms *Si request Si response Si crawler* and by assuming an average page size of 6KB, it can derive a simplified yet sufficiently exact version of the savings function.

$$S(N, CR, SF, FF) = N \times (1- CR \times SF \times FF) \times 6KB \ldots(29)$$

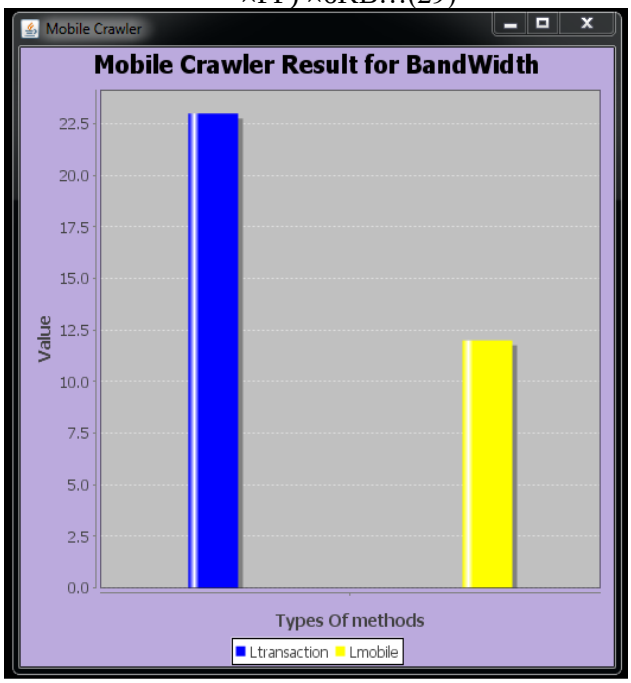


FIGURE 9: Bandwidth Consumption

V. JINI TECHNOLOGY

Mobile agents are ubiquitous in today's software applications from e-commerce to network management to data warehousing.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

Mobile agent developers implement these solutions in Java for several reasons: First and foremost, Java's built-in object-oriented language features are conducive to agent technology. Second, developers can be extremely productive using Java. Essentially, Java provides tools that simplify and expedite complex software development tasks.

JINI [11,12,13] is one such Java tool that allows us to build distributed applications with relative (and I emphasize relative) ease. This article introduces a mobile agent framework based on the Jini architecture. Jini provides a powerful tool in a Java developer's toolbox. Just as robots automate many aspects of manufacturing a computer, Jini automates and abstracts distributed applications' underlying details. These details include the low-level functionality (socket communication, synchronization, and so on) necessary to implement the high-level abstractions (such as service registration, discovery, and use) that Jini provides.

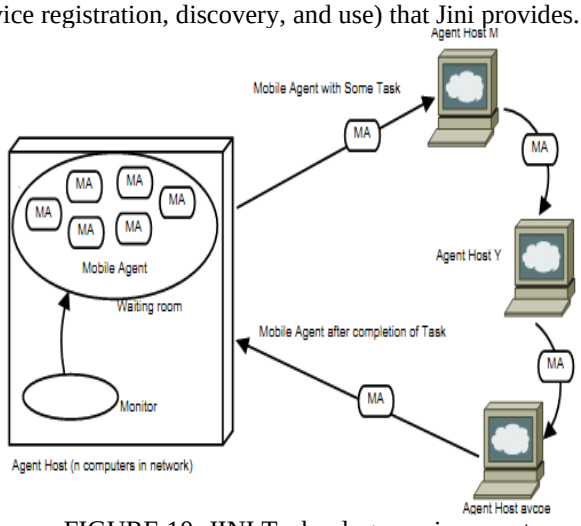
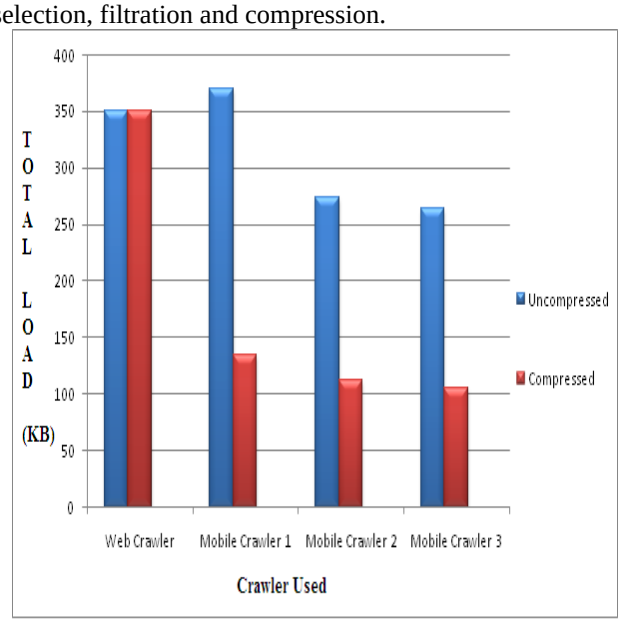


FIGURE 10: JINI Technology environment

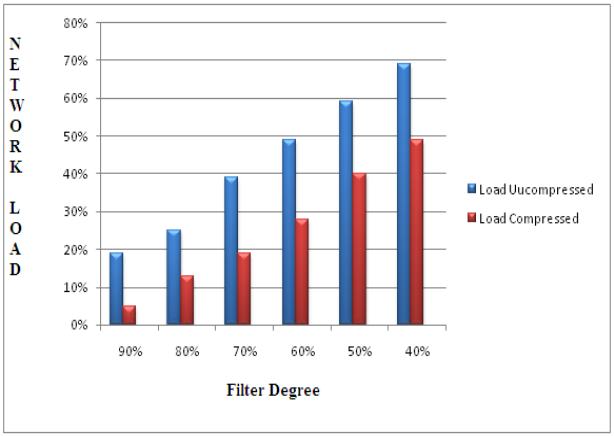
Mobile agent developed using JINI technology is a mobile agent which get transmitted from search engine (Agent) with users query to different web servers in search of information autonomously, complete the task assign to it and return to the search engine.

VI. RESULT ANALYSIS

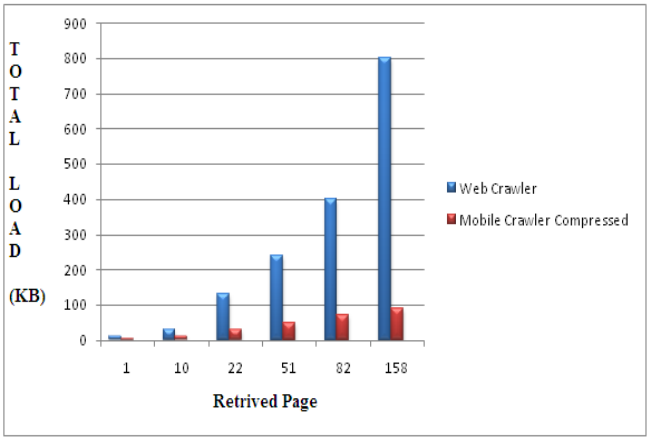
Following are the results of the experiment for remote page selection, filtration and compression.



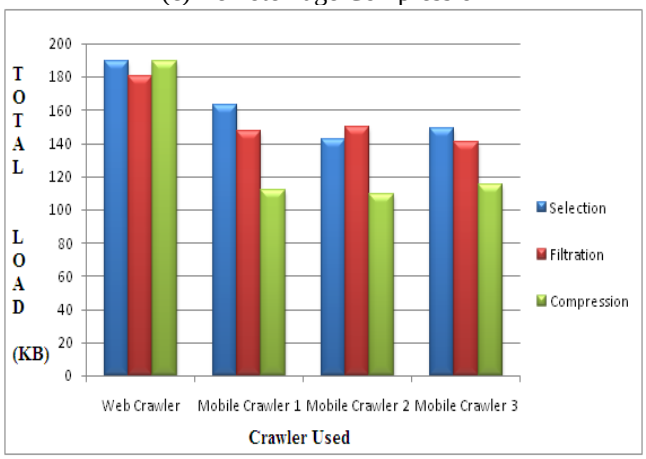
(a) Remote Page Selection



(b) Remote Page Filtration



(c) Remote Page Compression



(d) Combined Effect

FIGURE 11: Result Analysis

VII. CONCLUSION

We have introduced an alternative approach to Web crawling based on mobile crawlers. The proposed approach surpasses the centralized architecture of the current Web crawling systems by distributing the data retrieval process across the network. In particular, using mobile crawlers we are able to perform remote operations such as data analysis and data compression at the data source before the data is transmitted over the network. This allows for more intelligent crawling techniques and addresses the needs of applications, which are only interested in certain subsets of the available data.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

We have developed and implemented an application framework, which demonstrates our mobile Web crawling approach and allows applications to take advantage of mobile crawling.

REFERENCES

[1] Fiedler and J. Hammer. Using the Web Efficiently: Mobile Crawling. In Proc. Of the 7th Int'l Conf. of the Association of Management (AoM/IAoM) on Computer Science, San Diego, CA, pp. 324-329., August 1999.

[2] Bal, S. and Nath, R. Filtering the Non-modified pages at remote site during crawling using Mobile Crawlers. In Proc. of IEEE International Advance Computing Conference (IACC‟09),

[3] Ashutosh Dixit and Dr. A. K. Sharma,A Mathematical Model for Crawler Revisit Frequency"", 2010 IEEE 2nd International Advance Computing Conference.

[4] Satinder Bal and Rajender Nath, "A Novel Approach to Filter Non-Modified Pages at Remote Site without Downloading During Crawling", 2009 International Conference on Advances in Recent Technologies in Communication and Computing.

[5] Ashutosh Dixit and A.K Sharma, "Self Adjusting Refresh Time Based Architecture For Incremental Web Crawler", International Journal of Computer Science and Network Security (IJCSNS), Vol 8, No12, Dec 2008.

[6] L. Page and S. Brin, "The anatomy of a search engine", Proc. of the 7th International WWW Conference (WWW 98), Brisbane, Australia, April 14-18, 1998.

[7] Junghoo Cho and Hector Garcia-Molina.2000a."The evolution of the web and implications for an incremental crawler"., In Proceedings of the 26th International Conference on Very Large Databases.

[8] Cho, J., Garcia-Molina, H., Page, L., "Efficient Crawling: Through URL Ordering", Computer Science Department, Stanford University, Stanford, CA, USA, 1997.

[9] J Prasanna Kumar and P Govindarajulu, "Duplicate and Near Duplicate Documents Detection: A Review", European Journal of Scientific Research ISSN 1450-216X Vol.32 No.4(2009), pp.514-52 © Euro Journals Publishing, Inc. 2009.

[10] J. Fiedler and J. Hammer. "Using Mobile Crawlers to Search the Web Efficiently,"International Journal of Computer and Information Science, vol.1, no.1, pp.36-58,2000.

[11] [11] H. S. Nwana, "Software Agents: An Overview," Knowledge Engineering Review, Cambridge University Press, **11**:3, pp. 1996

[12] Complete code listings for the agent and agent host architecture components described in this article:

[13] [13] Gauri. S. Bhagat, M.S. Bewoor, Prof. Suhas H. Patil, "Improved Search Engine Using Cluster Ontology",IJAET,November,2011, ISSN-2231-1963

First Author R.G. Tambe is currently perusing Masters Degree in Computer Engineering. Completed Bachelor's degree from PREC, Loni, Maharashtra, India. Research area is Distributed and Parallel System, Mobile Agent System and Networking .

Second Author Prof. M.B. Vaidya is currently working as Associate professor in Amrutvahini College of Engineering, Sangamner, Maharashtra, India. Completed his Bachelor's degree in Computer Engineering from PREC, Loni, Maharashtra, India and completed his M-Tech degree from Walchand College of Engineering, Sangli. His total working experience in the field of education is of 18 years. His area of interest is Distributed and Parallel System, Applied Algorithm and Networking.