ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

A Novel Approach to Implement a Shopping Agent on a Scalable Web Crawler

Poonam Ghuli[1], Rajashree Shettar[2]

Assistant Professor, Department of CSE, R.V. College of Engineering, Bangalore, India [1]

Professor, Department of CSE, R.V. College of Engineering, Bangalore, India[2]

Abstract: Shopping Agent is a kind of Web application software that, when queried by the customer, provides him/her with the consolidated list of the information about all the retail products relating to a query from various e-commerce sites and resources. This helps customers to decide on the best site that provides nearest, cheapest and most reliable product that they desire to buy. This paper aims to develop a distributed crawler to help on-line shoppers to compare the prices of the requested products from different vendors and get the best deal at one place.

The crawling usually consumes large set of computer resources to process the vast amount of data in fat e-commerce servers in a real world scenario. So the alternative way is to use map-reduce paradigm to process large amount of data by forming Hadoop cluster of cheap commodity hardware. Therefore, this paper describes implementation of a shopping agent on a distributed web crawler using map-Reduce paradigm to crawl the web pages.

Keywords: Hadoop, map-reduce, shopping bots, Scalable web-crawler

I. INTRODUCTION

This is an era of Internet. With the incredible growth of information on the Internet, it's more and more difficult to find means to manage peta-bytes of data generated every day. Sorting 10s of tera-bytes of data on one node takes 2 and ½ days but a 100 node cluster will sort it in 35 minutes. Use of Fat-servers for processing such a huge amount of data implies high cost. So, today the trend is to use large number of cheap commodity nodes instead.

But large number of cheap nodes often led to failures. Thus, today one needs a new data-parallel programming model for forming clusters of commodity machines to process large amount of data generated every day. So, one of the solution is to use map-reduce framework such as Hadoop. In the recent years big leaders like Google, Yahoo have implemented hundreds of special-purpose computations that process large amounts of raw data, such as crawled documents, web request logs, etc., to compute various kinds of derived data, such as inverted indexes, various representations of the graph structure of web documents, summaries of the number of pages crawled per host, the set of most requested queries in a given day, etc. Most of such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues. As a reaction to this complexity, one has to design a new abstraction that allows one to express the simple computations that he/she is trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. These abstractions are inspired by the map and reduce primitive present in Lisp and many other functional languages. The map-reduce paradigm that uses concept of `cluster computing' is currently receiving considerable attention, both in the research and commercial arenas.

Cluster computing using networked commodity equipment has become an economic alternative for academic high performance computing facilities. Clusters aim to distribute compute intensive tasks over a set of back-end nodes on a network in order to speed up the processing, and hence reducing the turnover time for a solution to a complex problem.

Typically, a Shopping Agent works by sending out a crawler to fetch as many documents/products as possible from different e-commerce sites. This task of crawling documents is done using distributed computing methods (that is using the *Map-Reduce phase* of *Hadoop*). Another program, called an indexer, then reads these documents and creates an index based on the words contained in each document. The indexer uses a proprietary algorithm to create its indices such that, ideally, only the best matched, most meaningful and most popular results are returned for each query.

This paper follows an approach of building a crawler that crawls the actual e-commerce websites for their products and make a private database of all the products and their information from different e-commerce sites. This proposed work gets all its publicly accessible products of the

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

e-commerce sites from their own site-map that contains all the products resource locators for search engine optimization. We crawl on these site-maps for product resources and form a database of the information regarding each of their products.

II. LITERATURE SURVEY

Shopping Agents have been commercially developed and programmed ever since the advent of the e-commerce companies from 1990s. Also a lot of papers have been published about the ways to setup a full-fledged shopping agent that can be very reliable, high on performance and robustness and require as less consumer interaction with the actual vendor site as possible.

The basic concepts for building a ShopBot are very essential in the beginning. In the paper [1], authors provide us with the basics of the architecture and the strategies to be used in the building of a ShopBot. The paper included information about how the information about a particular product could be mined out from a site using methods of Heuristic Search, Pattern Matching and Inductive learning techniques instead of the usual technology of Natural Language Processing.

The work mentioned by M Eddahibi et al[2] is devoted to the design and the development of a shopping bot with the aim of overcoming the well-known difficulties in price comparison area. iShopBot is a shopping bot that combines several technologies, as semantic web, NLP, Multi-Agent systems and web data mining.

Facebook recently deployed Facebook Messages, its first ever user-facing application built on the Apache Hadoop platform[3]. Apache HBase is a database-like layer built on Hadoop designed to support billions of messages per day. This paper describes the reasons why Facebook chose Hadoop and HBase over other systems and discusses the application's requirements for consistency, availability, partition tolerance, data model and scalability. One can see observations for other companies who are contemplating a Hadoop-based solution over traditional sharded RDBMS deployments and how Hadoop could be used more in the core of the Shopping Agent application for maximum results.

For retrieving the data from other servers on the World Wide Web, we need to develop Web Crawlers [4]. Marc Seeger et al explains in detail the building blocks of a Web Crawler and how one must go about to program it. The architecture, design and even some part of the required implementation is discussed in the paper. The estimated and actual time is calculated as per the server bandwidth, database throughput, concurrent users etc. Also, the key factors tweaking by which the performance could be improved drastically are discussed and performed.

MapReduce[5] is a programming model and an associated implementation for processing and generating large data sets. Generally this model specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function which merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown by Jeffery Dean and et al in their paper.

The need to analyze structured data for various business intelligence applications [6] such as customer churn analysis, social network analysis etc., are well known. However, the potential size to which such data will scale in future will make solutions that revolve around data warehouses hard to scale. As data sizes grow the movement of data from the warehouse to archives becomes more frequent. In this paper, the authors present an active archival solution for data warehouses that makes use of Hadoop distributed file system (HDFS) to store the data in an always available and cost-effective manner. We use the method *column-store* discussed in this paper to store our crawled data using Hadoop.

DCrawler[7] is composed of several agents that autonomously coordinate their behavior in such a way that each of them scans its share of the web. An agent performs its task by running several threads, each dedicated to the visit of a single host. More precisely, each thread scans a single host using a breadth-first visit. Thus, dCrawler is distributed web crawler implemented using multi agents.

As one can see there are different implementation of shopping agents[8][9] using semantic web, multi agent systems, web data mining, NLP etc. But the proposed work uses an open source distributed framework called Hadoop to implement the crawler. Hadoop uses map-reduce paradigm which is based on master-slave concept. The crawler is responsible for collection of products and their prices from different e-commerce sites and registering them in database. The MapReduce paradigm is used for indexing part of data so that the retrieving of the search result becomes exponentially faster. Also, the PageRank Algorithm is implemented so that the most preferred search result is displayed at the top which is a dream for all the search providers.

III. PROPOSED WORK

The proposed work focus on implementing a Shopping Agent by accessing the sitemaps of e-commerce sites. To access the sitemap of a particular site by the proposed bot, the sitemap location must be specified using a directive in the *robots.txt file*. The proposed bot will try to find the following directive in the robots.txt file for a particular site:

Sitemap: http://yoursite.com/sitemap-location.xml

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

This helps search engines to recognize e-commerce sites better. Also, the sitemap provides access to a path where all documents of particular e-commerce sites are present. Thus, all their HTML pages are accessible to the consumers via the Search Engines like Google, Yahoo! etc. Therefore, if one uses a site *flipkart.com* as a registered e-commerce site then the sitemap of this site would be in a file known as robots.txt. One can access the sitemap by downloading the document *flipkart.com/robots.txt*. The file may further contain sublevels of sitemaps for different products according to the categorization done by the company web developers. The sublevels keep on going till we get the final HTML links to the actual products stored in the e-commerce sites. However, there are also sites that do not include and describe their sitemaps. Such sites cannot be accessed by our search bot.

Therefore, one can browse on sitemaps and get to the final HTML files that contain the information of actual products listed on respective sites. From these sites, the information can be obtained about the product like MRP (Maximum Retail Price), repository status (In Stock / Out of Stock / Imported) and their brand / author / publisher etc. Patterns between the product HTML links can be carefully extracted. Then use regular expressions to mine the information about the product from the webpage to the database created. This mined data can be shown to the consumers whenever they query for the product. Over the period of time the data on websites will be updated. There will be discrepancies between the data stored in database and the data currently present on the Website. The proposed work is based on assumption that the status and information of the product remain constant over some significant amount of time until the crawler crawls on those pages again.

A. Controller Module

This module is responsible for controlling the operations of the crawler. This enables user to enter the start URL, enter the maximum number of URL's to crawl, view the URL's that are being fetched.

B. Fetcher Module

This module starts by fetching the page according to the start URL specified by the user. The module goes to the World Wide Web, fetches the whole HTML page back to the system and applies the regular expression search to mine the data about the page. This process is the *crawling phase*.

C. Distributed Web Crawler

The data registered in this project include the title, the anchor text of the page and other back links that a page has to other pages on the Internet. The information about the product is therefore got in the *Retail Product Generator module*. The links so found are noted and are further crawled upon in the next pass according to the Breadth First traversal algorithm. These links are also recorded for the purpose of providing the pages their Page Rank.

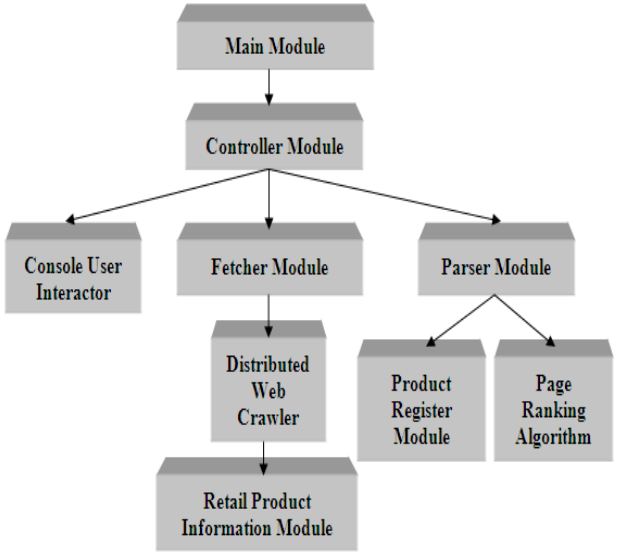

Fig 1. Overview of the system

D. Retail Product Generator module

This information together with the information of the last crawled page is sent together to the parser module. In case the HTML file of the current page to be crawled is too bulky in size, we scan only the limited part of the page due to hardware constraints and move on to crawl the next page in line. The fetcher module retrieves all the links in a particular page and continues doing that until the maximum number of URL's is reached.

E. Parser Module

This module parses the URL's fetched by the Fetcher module and saves the contents of those pages to the disk. The information submitted by the fetcher module is used here so as to actually save the page (offline search) and its contents onto the disk of the system. It is shown back to the user in results page when queried. The fetcher module also submits the information to determine the page rank of the last crawled page. Parser module is the one that does the actual calculation of the page rank of that page according to the *Page Rank algorithm.*

F. Page Rank algorithm

Therefore, here pages are indexed so that the most relevant pages get displayed on the top of the results page when the system is queried. It treats all the pages as HTML pages and saves it to the system with the information submitted about the page by the fetcher module and calculates the page rank for it. The calculation of page rank is done effortlessly using Map-Reduce paradigm. The information about the page is

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

mapped to individual systems in the cluster. The systems in the cluster then generate their individual page rank for the page and finally the NameNode reduces these bits of information submitted by the slave nodes and calculates the gross Page Rank for the page. The parser module is also responsible for the registering the individual products from the resource pages of the e-commerce sites which happens in the *Product Register module*.

G. Product Register module

The product and its other necessary information like brand / author / publisher, MRP, repository status (In Stock / Out Of Stock / Imported) and the resource locators are registered in this module in separate columns of the database. The parser module also has to take extra care about the database connections and be sure at all times not to open multiple connections, not to overburden the connection bandwidth, to take care of the possible exceptions that might occur as a result of any hardware or software malfunction.

IV. RESULTS AND ANALYSIS

Since the shopping agent uses Hadoop map-reduce concept, its performance lies in its cluster setup. i.e performance increases with increasing number of nodes in the cluster. The results are analyzed to see how the total running time of the task scales with the number of computing nodes. And it shows that the running time will be affected by varying the mapper/reducer assignments per node. The results are analyzed as follows.

A. Varying number of computing nodes

As shown in Fig. 2, the number of nodes in a hadoop cluster significantly affects running time of the processes. The analyzed result shows that the running time reduces from over 200 minutes to less than 30 minutes when the number of computing nodes is increased from 3 to 13. The overhead in managing mapper/reducer also affects performance.

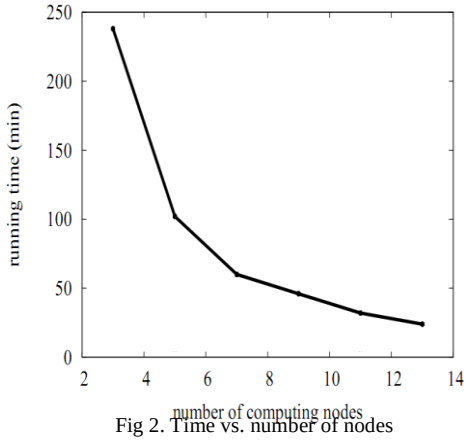

Fig 2. Time vs. number of nodes

B. Varying number of mappers/reducers per node

1) Varying number of reducers per node

Fig 3. shows the running time when the number of reducers per node is varied. Here the number of nodes and number of mappers per node are fixed. We find that running 3 reducers together on each of the node minimizes the total running time. Although, in a single job, increasing the number of reducers per node can reduces the time for the reduce stage, running time will be increased sharply in map and shuffle stage. As a result, the total running time will increase with the increase in number of reducers. This is because of the I/O bottleneck for concurrent access of data.

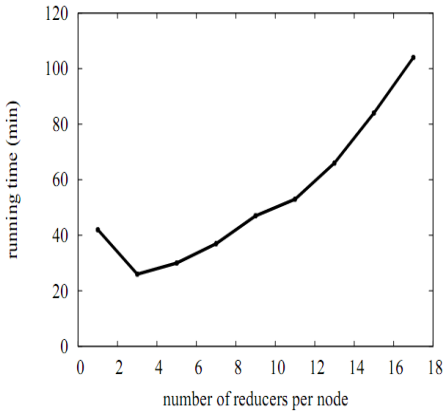

Fig 3. Time vs. number of reducers per node

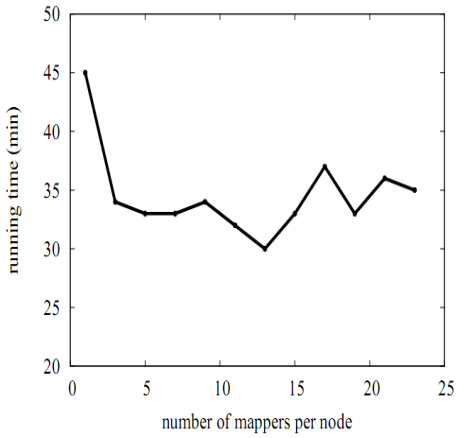

Fig 4. Time vs. number of mappers per node

2) Varying number of mappers per node

The fig 4 shows the total running time when the number of mappers per node is varied while fixing number of reducers as 7 for each node. We have found in our experiments that the performance is not affected by varying the number of mappers per node.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

V. CONCLUSION

This paper implements a specific application to get details about a product like price, ratings or any specification from all e-commerce websites at one place. The work focus on getting the cost details about books available at different e-commerece websites like Flipkart, Infibeam etc. This becomes convenient for the user to get details about different books and compare their prices from different websites at one place to get a better deal. This paper describes the implementation of a Shopping Agent in a distributed environment. During its implementation, we found that the crawling of the information on the e-commerce sites is tricky because of the inconsistency of placing of information about the product in the document and the denial of access permissions to those. Also, a distributed system crawling on the server was faster compared to the crawling done by the master system alone. The shopping agent will work more effectively and precisely if there was a way to link some sort of human interaction to the whole application. This could be done by giving ratings to the products or adding more e-commerce sites that the consumers find reliable and fast on delivery. Also reviews could be taken into consideration for the as a reference to other consumers who wish to buy a similar product.

REFERENCES

[1] Robert B. Doorenbos, Oren Etzioni, Daniel S, "A Scalable Comparison-Shopping Agent for the World-Wide Web", ACM 1997
[2] M Eddahibi, A Nejeoui and C Cherkaoui. Article: Towards an Intelligent and Deeply Automated Shopping Bot. International Journal of Computer Applications 45(24):21-27, May 2012. Published by Foundation of Computer Science, New York, USA.
[3] "Dhruba Borthakur, Jonathan Gray, Joydeep Sen Sarma, Kannan Muthukkaruppan, Nicolas Spiegelberg, Hairong Kuang, Karthik Ranganathan, Dmytro Molkov, Aravind Menon, Samuel Rash, Rodrigo Schmidt, Amitanand Aiyer, "Apache Hadoop Goes Realtime at Facebook", SIGMOD, 2011.
[4] Marc Seeger, "Building Blocks of a Scalable Web Crawler", Computer science and Media, 2010
[5] Jeffrey Dean, Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters", OSDI, 2004.
[6] Rajeev Gupta, Himanshu Gupta, Ullas Nambiar, Mukesh Mohania, "Efficiently querying archived data using Hadoop", CIKM, 2010.
[7] Sunil M Kumar and P.Neelima. Article: Design and Implementation of Scalable, Fully Distributed Web Crawler for a Web Search Engine. International Journal of Computer Applications 15(7):8–13, February 2011.
[8] Hyung Seok Lee, Dong Soo Jin, Jemi Choi, "Effects of Information Intermediary Functions of Comparison Shopping Sites on Customer Loyalty", Journal of Internet Banking and Commerce, August 2011.
[9] Khaled W. Sadeddin, "Online shopping bots for electronic commerce: The comparison of functionality and performance" Int. J. Electronic Business, 2002.
[10] Yun Wan, Satya Menon, Arkalgud Ramaprasad "A Classification of Product Comparison Agents" International Conference on Electronic Commerce '03, October 2003.