Supporting Distributed Search in Virtual Worlds

Hiep Luong, Dipesh Gautam, John Gauch, Susan Gauch, and Jacob Hendricks

University of Arkansas Fayetteville, AR 72701, U.S.A
{hluong,dgautam,jgauch,sgauch,jhendric}@uark.edu

Abstract. As three-dimensional (3D) environments become both more prevalent and more fragmented, the need for a data crawler and distributed search service will continue to grow. By increasing the visibility of content across virtual world servers in order to better collect and integrate the 3D data, we can also improve the efficiency and accuracy of crawling and searching by avoiding both the crawling of unchanged regions and the downloading unmodified objects that already exist in our collection. This helps to lower bandwidth usage during content collection and indexing, and for a fixed amount of bandwidth, maximizes the freshness of the collection. This paper presents a new services paradigm for virtual world crawler interaction that is co-operative and exploits information about 3D objects in the virtual world. By analyzing redundant information crawled from virtual worlds, our approach decreased the amount of data collected by crawlers, kept search engine collections up to date, and provided an efficient mechanism for collecting and searching information from multiple virtual worlds.

Keywords: Virtual World, Distributed Search, Data Crawling, Bandwidth.

1 Introduction

Search engines revolutionized the way we discover information in the World Wide Web. It seems natural that search would also vastly improve how we can discover and use information hidden in virtual worlds. Hence we have a need for virtual world crawlers, programs that automatically collect object data from virtual world servers. This object data is often text embedded in texture images, note cards, and chat messages that is triggered by avatar proximity or action. This means that a virtual world crawler must 'touch' the object before collecting its information. In other words, crawlers are expected to move around and approach objects in a region being explored. Traveling in a region, a crawler may encounter objects that have already been collected by previous crawls. When a crawler collects redundant object data, a crawler needlessly consumes bandwidth.

Our goal in this work was to explore a more efficient and exhaustive method of collecting content from virtual worlds. In this paper, we investigate the potential bandwidth savings that a collaborative crawling approach could achieve. In addition, this collaborative approach could be used to direct a crawler to a list of unvisited

A.A. Ozok and P. Zaphiris (Eds.): OCSC/HCII 2013, LNCS 8029, pp. 395–404, 2013.

regions or a region in the virtual world that has a high rate of change. We developed a focused crawler allowing us to collect data from Second Life and/or OpenSimulator virtual worlds. Once we gathered the data, we explored how frequently content changes in different regions and built a model of the rate of change in virtual worlds. With this model in mind, we have proposed an architecture that could allow the crawler to collect new or unvisited objects in a virtual world. Our empirical experiments using data from Second Life servers have shown that proper management of data redundancy based on our proposed architecture can decrease bandwidth traffic.

In this paper, we begin with a summary of related work on crawling data in virtual worlds. Then, we introduce our crawler architecture and present methods to evaluate data redundancy and bandwidth consumption savings during the crawling process. Next, we report our experimental results for this approach and discuss the impact of our work. The final section presents conclusions and discusses our ongoing and future work in this area.

2 Related Work

This section describes related work on crawling data in virtual world and efforts to support crawlers in exploring objects in 3D environment.

Crawlers for the WWW aim to collect exhaustive, fresh content from the WWW while minimizing bandwidth utilization. Cooperative crawlers incorporate methods that exploit information about web pages such as creation dates, update dates, file size and request frequency for each object of a website [2]. In [3], Chandramouli et al. designed a collaborative architecture in which web servers combine information from web logs and the file system to keep track of page creation, deletion, and modification. This information was available to web crawlers via a web service. They showed that, with this collaborative architecture, the crawler could discover new and valuable pages with reduced server traffic.

Interoperation among several virtual world environments remains a major challenge. The lack of interoperation on several current and possibly future virtual worlds is the main constraint on the growth of virtual worlds. Several researchers have been working to mitigate this constraint. Bell et al. [1] introduced VWRAP (Virtual World Region Agent Protocol) which addresses the problem of interoperability for a family of current and future virtual worlds, while [8] proposed an architecture and protocol for decentralizing multiuser virtual environments in which multiuser applications can exchange user agents and assets.

Among the current virtual worlds, Second Life[1] and OpenSimulator (OpenSim)[2] are the two most active worlds and have the most subscribers. There are two existing search services for Second Life and broader virtual worlds. The official Second Life search relies on the internal content database and does not extend to emulator worlds. The second service appears to rely on a combination of avatar crawlers and indirect database access for OpenSimulator worlds. OpenSimulator worlds can be connected

[1] http://secondlife.com/

[2] http://opensimulator.org/wiki/Main_Page

to by any Second Life client, but can also be hosted separately and combined into ad hoc grids to form separate virtual worlds.

There has been some work on exploring virtual worlds. Researchers have created a framework to collect avatar-related data using Linden Scripting Language (LSL) [10]. In [8], a crawler is used to collect spatial data of a user in Second Life in order to reveal relationships between behavior of real world humans and avatars in the virtual world.

Crawling data in virtual worlds is an essential task for the development of a distributed search service for 3D environments. Eno et al. ([4], [5] and [7]) demonstrated that virtual worlds could be effectively crawled with an autonomous agent that behaves like a normal human. They emulated a client protocol with an intelligent crawling agent to mimic normal user behavior. Their crawler navigates a region through an expanding spiral survey path[4]. In other work [6], Eno et al. examined landmarks and the picks to analyze the link between the regions. Their results showed that regions in the virtual world are linked similarly in pages of the flat web. Although they primarily studied regions within Second Life, they detected evidence of existence of a denser link structure on virtual world sites hosted in OpenSimulator.

In other work, Varvello et al. [9] analyzed Second Life's scalability, popularity, staleness of objects and quality of the user experience based on counting the objects in various regions at different points of time. However, staleness may not be accurately assessed merely by counting the number of objects. In our work, not only did we take the number of edited objects at different points of time into account, but we also compared the identities of the objects to get a count of unique objects. In this paper we focus on calculating the bandwidth required when regions are repeatedly crawled completely, including the collection of stale objects, and contrast that with the bandwidth needed by a collaborative architecture modeled on [3], a web service that publishes information about the virtual world's updates to the crawler.

3 Our Approach

Our goal is to estimate the amount of redundant data collected by a traditional crawler that repeatedly revisits regions and downloads all of the objects it can access. To do this, we developed a crawler that visited sample regions and analyzed the data collected, week by week, to identify any changes within that region. We then estimate the amount of data redundancy in this collection and describe a new architecture that would avoid the collection of redundant data, reducing bandwidth and the time necessary to update a search service's database. This section describes our virtual world crawler and the methods we used to identify redundant data as well as potential bandwidth consumption savings.

3.1 Overview

This section describes an overview of our proposed architecture that would support search across distributed virtual worlds by incorporating a collaborative crawler.

We have developed a search engine collection system that includes a knowledge base and indexing programs to store the collected data in addition to crawler software that gathers the content collection. The content collection is done by a set of virtual world client emulators interacting with the virtual world servers such as Second Life, OpenSim, etc. As the content is discovered, it is added to both a metadata database and an inverted index structure for query retrieval.

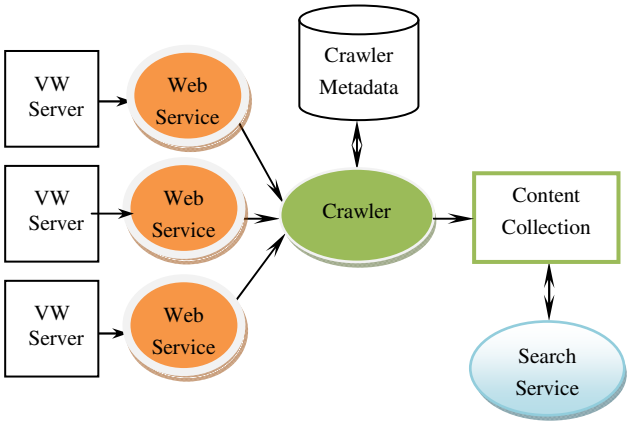

Fig. 1. Proposed System Architecture Overview

Unlike the web service paradigm proposed by [3] for flat web, we propose the development of a web service for each virtual world server such that each server shares object metadata in XML format with a collaborative crawler (Fig. 1.). The crawler then exploits this metadata information to identify the objects that have been added or modified since its last visit and focus on collecting only those objects. As another benefit, if the metadata also includes information about deleted objects, the crawler can remove those from the search engine's content collection, increasing the collection's accuracy. The crawler can also save metadata about the rate of change per region so that future crawls can target fast-changing regions more frequently.

As a first step to this, we developed a traditional crawler that visits regions and downloads all of the objects it can find. By repeatedly visiting sample regions, we develop a model of the rate of change of objects in different regions (additions, modifications, and deletions) so that we can estimate the potential benefits of our proposed architecture.

3.2 Collaborative Crawling in Virtual Worlds

We have developed collaborative crawler agents designed to collect user-generated content in Second Life and related virtual worlds. The crawling system architecture contains different components that are dedicated to crawling, coordination, and storage tasks. Specifically, the server manager starts the crawling tasks by assigning specific regions to crawl for individual agents. Then, it keeps track of completed and

queued regions as well as the status of the crawler in each region. The coordination layer consists of a server management program that coordinates individual crawlers and includes components for duplicate detection and queue management for the entire virtual world. Once the data is collected from each region, it is saved by the storage layer that includes both database storage and searchable index storage for the architecture [4].

When a crawler instance is assigned to explore data from a specific region, it will attempt to teleport to that region. When the crawler agent teleports successfully to a region, it begins storing object positions as they are automatically sent from the server. Scripts associated with objects are triggered when the crawler agent begins moving around the region to come into close proximity or touch the object.

Though more objects were discovered when the crawler agent moves near the object, the number of objects collected did not improved significantly with navigating the region when compared to the number of objects collected while standing at the landing area of the region. So, instead of navigating the region of interest, we sent the crawler to the landing area and let it to rotate around. For our preliminary experiment, we have selected 100 different regions from Second Life virtual world and let our crawler harvest data from these regions over several weeks. The data crawled were stored in the database.

3.3 Data Redundancy

The main issue for any information crawler is avoiding the collect of data that has already been downloaded. This creates unnecessary Internet traffic and wastes search engine resources during page collection and indexing.

In the virtual worlds, given the increased time necessary to physically navigate the 3D environment, it is particularly important to avoid collecting redundant data and to prioritize collecting the newly added objects or modified objects. We calculate the data redundancy collected by traditional crawling techniques by counting the common objects between two crawler collections, divided by the total number of unique objects collected by either crawl.

$$DR = \frac{UC}{TC} * 100$$

where UC is the number of unchanged objects which are crawled in a region, and TC is total number of objects crawled from that region. The average data redundancy is calculated using the average UC and TC values over all regions that are crawled.

3.4 Bandwidth Consumption Saving

In this section, we present a measure that estimates the potential savings in bandwidth consumed by the crawler using the proposed architecture. The bandwidth consumption savings metric is defined as follows:

$$BS = \frac{TC - UC}{TC} * 100$$

with UC and TC are defined in the above section. This bandwidth consumption saving value represents the percentage of savings that a crawler can achieve by avoiding the collection of redundant data.

4 Experiments

Our crawler has the task of interacting with the virtual world environment to collect content. It has been built using the OpenMV library available from Open Metaverse. This C# library provides the Second Life client emulation functionality necessary to connect to a Second Life server, move around, and interact with other objects or avatars. Multiple instances of the crawler connect to the virtual world at one time, coordinating their activities through the Server Controller. We have designed a test scenario that would determine the effectiveness of our crawler and help us characterize different classes of objects that are found in Second Life regions.

4.1 Data Collection

Our first task was to select regions to crawl. Initially, a list of regions is obtained by configuring our crawler to teleport to random Second Life regions from a seed region. From 300 regions selected from Second Life servers, we calculated the normalized number of objects crawled and the normalized time rate of collecting objects from each region (Fig. 2). The smallest regions took only a few minutes to be crawled while the largest regions required several hours to visit. Most of the regions among the 300 took from 6 to 12 minutes to crawl. In order to experiment with average sized regions, we excluded 100 regions including smallest and largest regions, and randomly selected half of the remaining 200 regions to obtain our final set of 100 regions.

In order to see how effectively the crawler captures objects in the virtual world, we launched the crawler with the same input parameters four times in four consecutive weeks. We started the crawler in a fixed time of the week (Thursday at 9:00AM) and saved data crawled for each week. Among 100 experimental regions, there were 3 regions from which the crawler was unable to collect objects; therefore we report our results over 97 regions. Table 1 shows a summary of data crawled over four weeks. For each snapshot, we report the total number of objects crawled. We then compare object records from consecutive crawls and calculate the number of objects added, deleted, modified and unchanged during each snapshot. Notice that the number of objects added and the number of objects deleted are relatively consistent from snapshot to snapshot, and are roughly 33% as large as the total number of objects crawled. The number of objects with modified attributes is also consistent from snapshot to snapshot, but these values are less than 2% of the total number of objects. Finally, the number of unchanged objects is roughly 66% of the number of objects crawled.

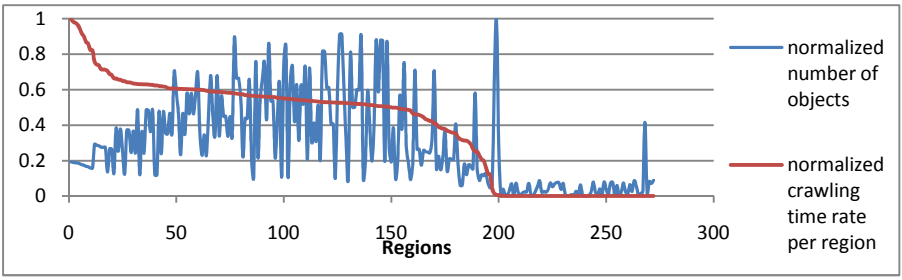

Fig. 2. Selection Second Life regions for experiments

Table 1. Data collection from 4 weeks

	#Objects crawled	#Objects added	#Objects deleted	#Objects modified	#Objects unchanged
Week 1 snapshot	490,850	0	0	0	0
Week 2 snapshot	486,294	157,492	162,048	5,627	323,175
Week 3 snapshot	481,313	154,860	159,841	8,074	318,379
Week 4 snapshot	449,729	136,653	168,237	8,293	304,783

4.2 Evaluation

Data Redundancy. Our goal with this experiment was to get a sense of how much redundant information is collected during each region crawl. Fig. 3 represents the data redundancy reported over four weeks. The DR_W2 curve shows the data redundancy between weeks 1 and 2. Similarly, the DR_W3 and DR_W4 curves show the redundancy between weeks 2 and 3, and between weeks 3 and 4. In all three cases, the redundancy values for each region have been sorted in decreasing order. Fig. 4 combines the data redundancy over four weeks, plotting the average data redundancy for each region crawled over the 4-week experiment. Notice that the median value for data redundancy is roughly 66%, which is consistent with the data in Table 1.

For each crawl, we also calculated the number of new objects that are created, the number of objects deleted, and the number of objects that are modified. In Table 2 we illustrate how the total number of objects changed is distributed in each week. TC_W2 is the total number of objects changed between week 1 and 2 for selected regions. Similarly, TC_W3 and TC_W4 are the total number of objects changed between week 2 and 3, and between week 3 and 4 respectively. We present the main percentile values at the 25th, 50th and 75th positions in order to see how the number of objects varies over four weeks.

Bandwidth Savings. A main advantage of our approach is its ability to analyze the data redundancy and decrease the amount of bandwidth used by crawlers. Fig. 5 represents the average bandwidth consumption saving for the whole data collection over four weeks.

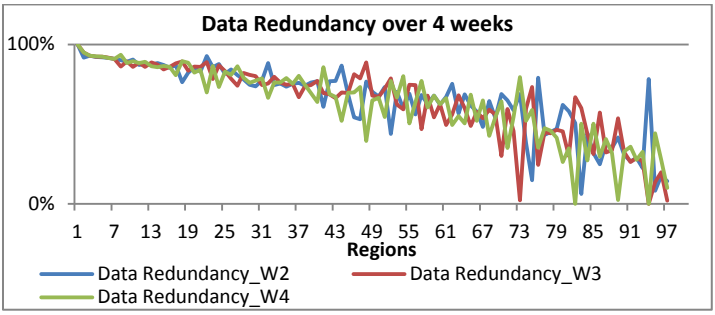

Fig. 3. Data Redundancy over 4 weeks

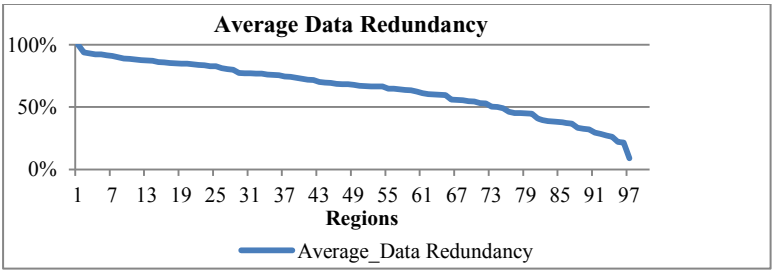

Fig. 4. Average Data Redundancy

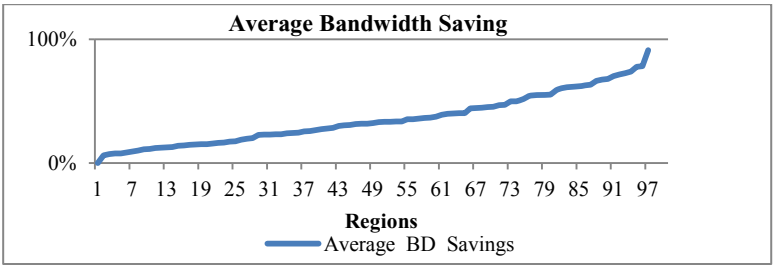

Fig. 5. Average Bandwidth Saving

Table 2. Percentile of number of objects changed over 4 weeks

Percentile	TC_W2	TC_W3	TC_W4
25th	4,138	4,119	4,072
50th	1,815	2,012	2,172
75th	1,324	1,213	1,287

5 Discussion

After investigating the data collected from our four weekly crawls of 100 regions of Second Life we have analyzed in detail the number of objects added, deleted and

modified in each region. Using this information, we calculated the bandwidth savings that would be provided by our proposed architecture. Table 3 shows a summary of the 25th, 50th, and 75th percentile values of bandwidth saving over four weeks. These values are very consistent from week to week. The bandwidth savings for the 25th percentile region averaged 48.50%, which indicates that roughly ½ of the objects in these regions remain unchanged from week to week. The 50th percentile region had an average bandwidth saving of 31.70%, which is a significant drop from the 25th percentile region. Finally, the 75th percentile region only had a bandwidth saving of 16.37%, which is only 1/3 of the 25th percentile value.

Table 3. Percentile of bandwidth savings over 4 weeks

Percentile	Saving Week 2	Saving Week 3	Saving Week 4	Average Saving
25th	45.86%	46.31%	49.70%	48.50%
50th	29.61%	30.14%	31.45%	31.70%
75th	15.16%	14.89%	16.50%	16.37%

According to our investigation, the crawler seemed to collect redundant data over the weeks because many of the objects in a virtual world, objects like chairs, trees, towers and buildings, are static. On the other hand, the crawler failed to collect certain data, particularly data pertaining to dynamic objects. Dynamic content such as sand boxes, avatar outfits, billboards and some robots may appear at a place and time only to disappear at another time. Similarly, when the crawler moves, some objects are missed because they are far away from the crawler avatar, not in the range of visibility. Of course another reason that the crawler failed to collect data about some object is that the object has actually been removed from the region.

6 Conclusions

The goal of our research was to implement and validate an intelligent crawler that collects data from virtual worlds. We have demonstrated that crawler performance can be significantly enhanced in terms of bandwidth consumption savings. Our approach to reduce bandwidth usage was to avoid redundant object collection.

We have shown that there is typically a considerable amount of data redundancy in crawling virtual worlds. This can lead to unnecessary bandwidth usage if this redundant data is collected by a crawler. Our approach was empirically tested using data we collected from Second Life servers that contain different kinds of objects in virtual worlds. The experimental results showed that our approach's ability to analyze the data redundancy in crawling potentially helps to reduce resource consumption of the collection process by downloading only unvisited and newly added content.

Our future work includes research into appropriately weighting objects and using interactive content that could guide the crawler to collect more useful object data, while avoiding previously collected objects. We also plan to extend the search function with flexible input parameters to create search service more appropriate for virtual world environments.

Acknowledgements. This research is supported by the NSF grant number 1050801 - III: EAGER: Mapping Three-Dimensional Virtual Worlds.

References

1. Bell, J., Dinova, M., Levine, D.: VWRAP for virtual worlds interoperability [Standards]. IEEE Internet Computing 14(1), 73–77 (2010)
2. Buzzi, M.: Cooperative Crawling. In: Proceedings of First Latin American Web Congress (LA-WEB 2003), pp. 209–211. IEEE Computer Society, Washington, DC (2003)
3. Chandramouli, A.: A co-operative web services paradigm for supporting crawlers. Ph.D. dissertation, Univ. of Kansa, Lawrence, KS, USA (2007)
4. Eno, J.: An Intelligent Crawler For A Virtual World. Ph.D. dissertation, Univ. of Arkansas, Fayetteville, AR, USA (2010)
5. Eno, J., Gauch, S., Thompson, C.: Intelligent Crawling in Virtual Worlds. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2009), vol. 3, pp. 555–558. IEEE Computer Society, Washington, DC (2009)
6. Eno, J., Gauch, S., Thompson, C.: Linking Behavior in a Virtual World Environment. In: Proceedings of the 15th International Conference on Web 3D Technology (Web3D 2010), pp. 157–164. ACM, New York (2010)
7. Eno, J., Gauch, S., Thompson, C.: Searching for the Metaverse. In: Spencer, S.N. (ed.) Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology (VRST 2009), pp. 223–226. ACM, New York (2009)
8. La, C.A., Michiardi, P.: Characterizing User Mobility in Second Life. In: Proceedings of the First Workshop on Online Social Networks (WOSN 2008), pp. 79–84. ACM, New York (2008)
9. Varvello, M., Picconi, F., Diot, C., Biersack, E.: Is there life in Second Life? In: Proceedings of the 2008 ACM CoNEXT Conference (CoNEXT 2008), Article 1, p. 12. ACM, New York (2008)
10. Yee, N., Bailenson, J.N.: A method for longitudinal behavioral data collection in second life. Presence: Teleoper. Virtual Environ. 17(6), 594–596 (2008)