9-4-2010

Can a Scalable System be Built to Capture Web Page Thumbnails?

Feras Hirzalla
Purdue University, hirzalla@gmail.com

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Feras Hirzalla

Entitled Can a Scalable System be Built to Capture Web Page Thumbnails?

For the degree of Master of Science

Is approved by the final examining committee:

John Springer
Chair

Brandeis Marshall

Eric Matson

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): John Springer

Approved by: Gary Bertoline 08/24/2010
Head of the Graduate Program Date

Graduate School Form 20
(Revised 1/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:
Can a Scalable System be Built to Capture Web Page Thumbnails?

For the degree of Master of Science

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Teaching, Research, and Outreach Policy on Research Misconduct (VIII.3.1)*, October 1, 2008.*

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Feras Hirzalla
Printed Name and Signature of Candidate

8/23/2010
Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/viii_3_1.html

CAN A SCALABLE SYSTEM BE BUILT TO CAPTURE WEB PAGE THUMBNAILS?

A Thesis

Submitted to the Faculty

of

Purdue University

by

Feras Hirzalla

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

December 2010

Purdue University

West Lafayette, Indiana

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Hirzalla, Feras M. M.S., Purdue University, December 2010, Can a Scalable System be Built to Capture Web Page Thumbnails? Major Professor: John Springer.

Screenshots of web pages are a novel user interface improvement that can be used in various systems. Several software packages can be used to generate a visual representation of a web page, but none of these packages are designed to capture web pages in a scalable way.

The goal of this research is to build and test a web page thumbnailer that can simultaneously capture screenshots of multiple web pages using multiple threads, thus maximizing the throughput of web pages that can be processed per unit of time. From the data collected during tests, thread count usage recommendations can be made.

To test the thumbnailer, the author used a sample set of 30,000 web pages. Web page sizes were collected from the sample set, and 10,000 web pages were assigned to small, medium, and large categories. The thumbnailer was timed for different thread counts for each of the size categories. The data showed that as more capture threads were added, the time it took to generate visual representations of web pages decreased.

CHAPTER 1. INTRODUCTION

This chapter introduces the research and provides the scope, significance, and research question that will be addressed. Assumptions, limitations, and delimitations of this research will also be explored.

1.1. Background

People of all ages and backgrounds use the World Wide Web on a daily basis. Many of these users come from a wide range of non-technical disciplines where there is minimal interaction with computers. With the rise of Graphical User Interfaces (GUI) in operating systems, applications have become more accessible and simple to use. As the Web grew, user interfaces in browsers and search engines evolved to be more graphical and included visual aspects that identify content. Every Website on the internet has its own distinct look and feel; this led some researchers to experiment in utilizing the distinct Website feel by supplementing user interfaces with screen captures of Websites. Similar to other systems on the Web, the process of generating screen captures that can be converted into web page thumbnails must be scalable and reliable.

While scaling most Web technologies has been extensively researched, generating thumbnails of web pages has been overlooked. Many software packages can produce thumbnail screen captures of web pages, but this software was not designed to capture thousands of thumbnails at one point in time. Can a scalable system be built to generate web page thumbnails concurrently? Can this system be architected in a way where the thumbnail throughput increases as more capture threads are increased? This research examines these areas and provides a methodology that can be used to test the newly developed system.

1.2. Research Question

The main question this research addresses is:

- Can a scalable system be built to capture web page thumbnails in a concurrent manner?

1.3. Scope & Contribution

Producing thumbnail screen captures of web pages touches several areas of software development. It is difficult to categorize these areas due to overlaps but they mainly consist of image processing, Web protocols such as HTTP, and screen scraping of web pages.

The method shown outlines the steps in scaling the process of web page thumbnail generation. Scalability is the ability of a system to process growing amounts of data gracefully and be able to grow as required (Bondi, 2000). To achieve scalability, a system was designed that can process web pages concurrently and scale out by increasing the number of simultaneous capture threads ensuring an increased thumbnail generation throughput. Throughout the testing phases, this system was benchmarked using Windows Performance Monitor (Perfmon) to note hardware limitations.

Rather than focusing on image processing algorithms, screen scraping, or the actual methods of thumbnailing, the Internet Explorer rendering engine was used to view the web pages. The research focuses on designing and providing a proof of concept implementation of a web page thumbnail generation system that emphasizes parallel processing, and methods of testing the system.

1.4. Significance

Internet World Stats (2009) estimates that there are close to 1.7 billion users on the Internet. Online services must be able to serve thousands of requests and grow as demand for these services increases. To cope with the increasing demand, Website developers have set out to find innovative ways to lay out information with the goal of simplifying and improving the online experience for visitors. web page thumbnails are a user interface improvement that can increase Website revisitation, ease browsing, and teach users how to navigate their history and the hierarchy of web pages efficiently (Cockburn, Greenberg, Jones, McKenzie, & Moyle, 2003). Many other uses of web page thumbnails will be explored in the literature review.

The rise of web page thumbnails as a user interface improvement along with the sheer size of the World Wide Web necessitates that the thumbnail generation process be able to scale to cope with the increasing demand. Additionally, the thumbnail capture system will eliminate a hurdle for developers that would like to incorporate web page thumbnails in their applications to simplify interfaces and increase accessibility.

1.5. Assumptions

The assumptions for this research include:

- When matching a web page to an original thumbnail:
    - Images with slight color variation due to resizing and processing will not count as an incorrect thumbnail.
    - Changes in actual content of a web page will not count as an incorrect thumbnail.
- The thumbnailer cannot reach login protected pages.
- All web pages in the sample set are publically accessible on the Internet.
- Because the sample set includes a diverse set of public Websites, the assumption is that they will be available and responsive.

1.6. Limitations

There are several limitations for this research:

- The final implementation is implemented using Microsoft technologies and therefore the system is not cross platform.
- The network connection used in the study is not a commercial connection, thus the system might not ever have access to the full 10 megabits the connection is rated at. This means that the saturation point of the network has changed between trials.

1.7. Delimitations

The research was conducted with the following delimitations:

- The study focused solely on the scaling process and concurrency. Concurrency refers to processing multiple web pages at the same time. Internet Explorer was used to render the web pages before generating screen captures and converting them to thumbnails.
- The experiments were conducted with 10 trials for each size group and thread count. This can be considered a small number of trials for a quantitative study. This study aims to provide a proof of concept implementation of a system that can scale the thumbnail generation process. By using a multi-threaded architecture, thumbnails of multiple web pages can be saved simultaneously and served to requesting end users. Future research can be conducted on larger sample sets with more trials to investigate whether or not the results can be extrapolated to the entire World Wide Web.
- Although the initial sample set collection excluded pages that returned an HTTP 404 Not Found error message, Web servers may sometimes return this error response during thumbnail capture. In this study, the thumbnailer was not able to detect error responses and adjust the sample size

accordingly. Future research can address this edge case by dynamically adjusting the sample size based on the number of error responses.

1.8. Definitions

Benchmark: A performance test of hardware and/or software (Computer Definitions, n.d.). The system was benchmarked during each of the experiments to develop a system profile between thread counts.

Concurrent: Happening at the same time as something else (The American Heritage® Dictionary of the English Language, 2003). In this context, concurrency is processing more than one web page at the same time.

Resource Description Framework (RDF): A standard model for data interchange on the Web. RDF extends the linking structure of the Web and uses URIs to represent relationships between things (W3C RDF Group, 2004). The Open Directory provides RDF files that represent the directory's content. In this study, the sample set was collected by parsing these files.

Scalable: The ability of a system to process growing amounts of data gracefully and be able to grow as required (Bondi, 2000).

Thread: A portion of a program that can run independently of and concurrently with other portions of the program (The American Heritage® Dictionary of the English Language, 2003). The author has chosen to develop a multi-threaded application as a method of scaling the web page thumbnailer by capturing each web page in a separate thread and running multiple threads simultaneously.

1.9. Summary

This section introduced the background of the research and set limits on the scope of the research area. Key terms were defined and specific limitations, delimitations, and assumptions were stated.

CHAPTER 2. LITERATURE REVIEW

In this section, various literature will be reviewed in relation to how these thumbnails can be beneficial in supplementing user interfaces, how they're generated, and how similar systems such as Web crawlers scale to millions of web pages.

2.1. Literature Review Approach

This literature review will begin by outlining how web page thumbnails have been used to supplement user interfaces. While these thumbnails have been explored and used for years, the area of scaling the thumbnail generation process remains unexplored in academic work. This review will begin by outlining the benefits and previous work done with web page thumbnails, and then describe different aspects of the thumbnail generation process. Some of the methods outlined might touch on related areas such as Web crawling, image processing, and various other World Wide Web related disciplines where scalability issues were tackled.

2.2. Web Page Thumbnails as User Interface Aids

Researchers have used web page thumbnails in various applications such as mobile Web browsers, history and bookmarks, search engines, revisitation, and easy web page identification.

Cockburn, Greenberg, Jones, McKenzie, and Moyle (2003) researched methods on improving user web page revisitation. According to Cockburn et al. (2003), the average user revisitation rate for a study conducted was between

61% and 92% where the mean number of distinct URLs visited was 1227. To make revisitation simpler, the researchers proposed that web page thumbnails be used in browser history and bookmark lists with a system called WebView. Several participants were surveyed on a regular Netscape browser and one that utilized WebView; the overwhelming response to the thumbnails was that they were only marginally useful for web pages they already recognized but that they would be much more useful in visiting unfamiliar web pages. Czerwinski, Dantzich, Robertson, and Hoffman (1998) had similar findings when nine subjects with ages ranging from 18 to 50 years old were shown a set of 100 web page snapshots during a test session, and then brought in four months later for another test session. With web page thumbnail captures visible, the times it took for users to recognize the pages were comparable with the values collected four months earlier. Initially, the subjects' ability to find the pages was significantly hampered when the thumbnails were taken away but improved as the subjects conducted more trials.

PadPrints addressed browser history revisitation by providing a tree diagram of a user's path through the Web with thumbnails attached to each node (Hightower, Ring, Helfman, Bederson, & Hollan, 1995)***;*** users were able to navigate back to web pages in their history 61% faster than users not using PadPrints**.** Additionally, Teevan, Cutrell, Fisher, Drucker, Ramos, Andre, and Hu (2009) conducted a two-phased study where web page revisitation was explored. Participants, ages ranging from 18-65, came from different departments at a large software company participated. The first phase was completed by 276 participants where they were tasked with finding new pages on the Web. Of the 276 participants, 197 participants went on to complete phase two where they were tasked with revisiting the web pages found in the first phase. Each participant saw different representations of web pages, these include visual snippets (text and thumbnail representation), thumbnails, and plain text snippets. The study found that participants were able to refind certain search results the fastest when a thumbnail was used to supplement the plain text representation of a web page, followed by thumbnails only, then plain text representations. Kaasten and Greenberg (2001) used thumbnails to improve page recognition in an experiment where back buttons, history, and bookmark thumbnails were integrated in a Web browser.

Chen, Ma, and Zhang (2003) showed that by generating thumbnails of web pages and incorporating some clever HTML parsing, they could scale regular web pages to mobile device browsers and provide a better browsing experience on small screen devices. Thumbnails are generated on the fly when a user requests a web page, and a layer of HTML enables links and facilitates user interaction. The use of web page thumbnails as a user interface aid is not limited to small screen devices, Yoo, Lea, and Kim (2008) developed a browser aimed to improve web page browsing on large televisions. The browser addressed the difficulty end users have in pinpointing hypertext links from 10-feet away by attaching zoomable thumbnails to hyperlinks. Task completion time and user convenience were measured for 10 trials, subjects were able to complete the news search task assigned with and without thumbnails in 5.9 seconds and 12.8 seconds respectively.

Metadata aggregation services have also benefited from thumbnail generation. In a project at the University of Illinois at Urbana-Champaign (UIUC), thumbnails were automatically generated for Web resources referenced in the university's Committee on Institutional Cooperation (CIC) metadata portal to improve accessibility (Foulonneau, Habing, & Cole, 2006).

2.3. Generating Thumbnails

Generating web page thumbnails is not a simple process. Several decisions must be made on thumbnail size, storage requirements, image formats, and what parts of a page to capture. There are many visual attributes that need to be considered when generating thumbnails. Colors, legible text, distinctive images, and web page layouts are all visual identifiers observed by users identifying a thumbnail (Kaasten, Greenberg, & Edwards, 2001)***.*** The authors showed that 90% of 100x100 or larger thumbnails were primarily identified by the text on the web page; color and layout based identifications were more predominant in thumbnails of 96x96 or smaller because each web page has its own individual look. One aspect of this unique look of web pages is the width of the web page; while some web pages employ fluid layouts, others have fixed widths that can range from 800 pixels to 2500 pixels. According to the World Wide Web Consortium (n.d.), as of January 2010, 96% of Web users have a display resolution of 1024x768 or greater. The web page capture area takes into account current resolution statistics to maintain consistency between how users see web pages in their browsers and what the system captures.

Another consideration is the image compression format used when generating thumbnails. There are countless image formats from which to choose from. These mainly fall into two categories, lossy and lossless image formats. The difference between the two categories is that lossy image formats lose quality after repeatedly compressing and decompressing a file, whereas in lossless image formats, the exact original data can be reconstructed from the compressed data (Nationmaster.com, n.d.). The World Wide Web Consortium (n.d.) recommends Portable Network Graphics (PNG) for use in still graphics as a replacement for GIF. PNG is a lossless image format that provides good quality images with small file sizes (World Wide Web Consortium, 2003) proving suitable for a scalable system that would store a large amount of web page thumbnails.

2.4. Providing Reliable Thumbnails

Reliability in the context of web page thumbnails refers to the process of generating clear, viewable images of varying types of web pages. Generating thumbnails on a list of predefined web pages might have been a trivial task during the early days of the Web, but the advent of Asynchronous Javascript and XML (AJAX) and Flash make it more difficult to achieve reliable thumbnails due to delayed loading and larger file sizes. A capture delay was incorporated to allow pages that rely on client-side scripts to render correctly (Foulonneau et al., 2006). An HTTP HEAD request was issued before capturing a thumbnail to retrieve the original URL in the event of any redirects (Foulonneau et al., 2006).

2.5. Scalable System Architecture

To the author's knowledge, there has been no research conducted on scaling a system that generates web page thumbnails. Researching scalable systems led the author to examine search engines, specifically Web crawlers due to their ability to scale to millions of web pages at a time. Boldi, Codenotti, Santini, and Vigna (2004) define scalability as “[t]he number of pages crawled per second and agent should be (almost) independent of the number of agents, in other words, we expect the throughput to grow linearly with the number of agents” (p. 3). Most Web crawlers also have a URL queue where URLs are prepared for crawl; caching, duplicate URL replacement, and other techniques can be applied to the URL queue to increase performance (Broder, Najork, & Wiener, 2003).

To create a scalable system, concurrent programming techniques utilizing multiple threads were employed. As mentioned earlier, the goal is to scale up the number of web pages undergoing processing as the number of agents increases. Most modern languages such as C# or Java, or even some of the newer concurrent languages such as ErLang can attain this goal.

Compared to Web crawling, the thumbnail generation process has less modules to manage. Web crawlers are actively crawling content they encounter by following hyperlinks throughout the Internet and passing this content to other modules with specific tasks. For example, a parsing module is used to clean up the HTML and extract keywords from the document. However, the thumbnail generation process is more of an on-demand process where a thumbnail is requested, queued up, generated, and saved to the file system. These thumbnails are cached for future access and served up when requested. Moving on to the next part of the literature review, data accessibility and system interaction are discussed.

2.6. Data Access & Web Services

A system was created that allows services and users to request thumbnails for specific URLs through a Web service layer (Curbera, Duftler, Khalaf, Nagy, Mukhi, & Weerawarana, 2002). This Web service layer will interact with an agent that serves up thumbnails from the cache or appends the web page to the URL queue for processing. Over a period of time, the system will build up a large cache of thumbnails for the most commonly requested web pages. Because these web pages are all publically accessible on the internet, Copyrights are not expected to be an issue because all web pages captured are publically accessible. Additionally, full web page thumbnails ensure that an organization or individual's logo and branding are intact with the content.

2.7. Updating Thumbnails

As stated above, thumbnails are stored after being retrieved once; a challenge is keeping these cached images up-to-date. Because the system operates on an on-demand basis, any idle processing time is used to update existing cached versions of thumbnails. The SETI@home project adopted a similar approach where clients only participate in the public computing when in an idle state (Anderson, Cobb, Korpela, Lebofsky, & Werthimer, 2002). The priority of updating web pages is determined by the URL queue, mainly by sorting web pages in ascending order on the file last updated value.

2.8. Summary

This concludes the literature review. While there isn't a large amount of research in the area of generating web page thumbnails, there is plenty of research on how these thumbnails were used to improve user interfaces. Most of the research in this area had qualitative aspects to investigate how thumbnails influenced user behavior. Combining some of the stated approaches from the Web crawler world and focusing on scalability in the initial design process was valuable in developing and testing this system.

CHAPTER 3. METHODOLOGY

This chapter will introduce the system and research framework, and then provide methods on how the system was tested. The data collection approach will be outlined and then the chapter will end with methods of analyzing the collected data.

3.1. Research Framework

This research uses an experimental quantitative method to evaluate the scalability and reliability of software written by the author to capture web page thumbnails. The thesis answers the initial research question of whether or not a scalable system can be built to capture web page thumbnails, and more importantly, document how this system was tested.

The author has chosen concurrency as a means of producing a scalable system. Although concurrency means that things are happening at the same time, the hardware reality tells a different story. When processing instructions, CPUs allocate time slices to different threads based on priority (MSDN, 2010b). For single core CPUs, this means that processing isn't technically happening at the same time. Developing a multi-threaded application to process web pages minimizes processor idle time ensuring a higher throughput of captured thumbnails. From this, the hypotheses are:

H_0: *As the number of simultaneous capture threads increases for a fixed number of web pages, the thumbnailer's process duration does not decrease.*

H_a: *As the number of simultaneous capture threads increases for a fixed number of web pages, the thumbnailer's process duration decreases.*

Several variables were measured during the experiments to test the hypotheses and ultimately answer the research question of whether or not a scalable system can be built to capture web page thumbnails.

The experiments include the following independent variables:

1. The number of threads allowed to run on the test system during each trial.
2. Number of web pages to capture (Section 3.4)
3. A categorization of web pages based on their size in kilobytes.

The experiment will include the following dependent variable:

1. The time it takes to generate thumbnails of web pages in the sample set for the chosen size category.

The time it takes the thumbnailer to process the sample set of web pages heavily depends on the hardware capacity. These experiments were conducted on a desktop computer with the following specifications:

1. Intel Core 2 Quad CPU at 2.33GHZ
2. 8.00GB of RAM
3. 500GB 7200RPM Hard drive
4. Intel integrated video
5. Windows 7 Home Premium - 64 bit
6. Up to 10 Megabit Cable

3.2. Sample Set & Data Collection

The list of web pages used to test the system was obtained from the Open Directory Project, which publishes their data using the Resource Description Framework (RDF). The directory is the largest, most comprehensive human-edited directory of the Web (Netscape, 2010). More importantly, web pages in the directory have a diverse set of technologies that can affect the reliability of the thumbnailer such as Flash, Asynchronous Javascript & XML, and differing amounts of content.

To test the software's scalability, a subset of the directory was analyzed and 10,000 web pages of small, medium, and large sizes were extracted producing a full sample set of 30,000 web pages. The specific limits of each size were determined by analyzing the dataset. To minimize variation in the results, 10 trials were conducted where the thumbnailer was timed until completion for each size category and thread count in the sample set.

The experiments began by processing the sample set using 1 thread. The thread counts were then increased until hardware limits were observed based on the system profile. The experiments began with incrementing thread counts by 10 for each execution, then increasing these thread increments by 25-50 threads in the higher ranges. The duration was plotted against the number of threads and trend lines were fitted by examining the square of the correlation coefficient (r^2) value closest to one to achieve the highest linear reliability (Clemson, n.d.). The average of the 10 trials was used for each size group and thread count.

To monitor the hardware capacity, the system was profiled as the thumbnailer processed the list of web pages. Windows Performance Monitor (Perfmon) was used to profile the system. The following system variables were tracked:

- Memory usage
- CPU Usage
- Disk access
- Network Utilization

Based on the system variables above, various Perfmon counters were chosen to collect data on the system. The appendix contains a full list of counters tracked with their respective definitions, the following counters were of most interest to this research:

1. Avg. Disk sec/read: Indicates the average time it takes to read data from the disk in seconds. The disk read response time is a critical measurement in determining whether or not disk slowdowns resulted in a

slower thumbnailer. According to Microsoft, the average value should be below 10 ms with spikes not being higher than 50 ms.

2. Avg. Disk sec/write: Indicates the average time it takes to write data from the disk in seconds. Similar to Avg. Disk sec/read, this counter helps determine if a slow disk response decreases the thumbnailer's efficiency. The average value should be below 10 ms and spikes should be no higher than 50 ms (Microsoft TechNet, 2005a).
3. Memory pages/sec: Indicates the rate at which pages are read from or written to disk. Microsoft TechNet (2005b) states that this value should be below 1,000 at all times.
4. Memory Pages Input/sec: The rate at which pages are read from the disk to resolve hard page faults.
5. Network Utilization in Megabits: Measured by collecting the total amount of bytes transferred over the network adapter and converting it to megabits. This indicates whether or not the network connection is the source of a bottleneck for the thumbnailer.

As for thumbnailer reliability, this was a manual process. The reliability of an individual thumbnail is binary; that is, the system either captures the correct web page thumbnail or it doesn't. To determine this, the author periodically checked the thumbnails captured and identified any issues throughout the trials. The original web page and the thumbnail had to be identical for it to be counted as a correct thumbnail.

3.3. Technology Overview

This section will give an overview of the methods used in different parts of the research to collect sample size data, system performance data, and will provide an overview of the thumbnailer's architecture.

3.3.1. Sample Set Size Collection

The Open Directory Project was used to gather a sample set of web pages. Approximately 4.3 million web pages were imported into Microsoft SQL Server by parsing the downloadable RDF files (Netscape, 2010). A random subset of 30,000 web pages were extracted by assigning a unique identifier to each record using Microsoft SQL Server's *newid()* function, then sorting the records on the unique identifier in ascending order by size (MSDN, 2010a). The size collection had to include a web page's HTML and any associated objects required to render the page. This includes Javascript, CSS, and any images displayed on the page.

Several approaches were considered for the size collection. The first attempted strategy was to issue an HTTP HEAD command to the Web servers and examine the HTTP Content-Length header of the page. HTTP HEAD is an HTTP verb identical to HTTP GET, except that the server does not return a message body as part of the response. The HTTP Content-Length header provides the size of the requested entity's body (W3C, 1999). To examine the size of other elements on the page, the system would have to download the document's HTML, parse it, and issue HTTP HEAD commands to any linked Javascript, CSS, and image files on the page. However, this approach is problematic and often produced invalid results because HTTP headers can be manually assigned by Web servers and dynamic languages such as PHP.

Instead of using the HTTP Content-Length header field, a custom script was written to visit each web page and download the HTML content and any associated objects to the local file system. Wget, a tool that ships with the GNU operating system, was used to download the content. Wget is a package designed to retrieve files using HTTP, HTTPS, and FTP (Free Software Foundation, 2003). Wget provides various benefits for the web page downloading process. Firstly, it has the ability to save a single page, or a full Website. Secondly, versions of Wget exist for Windows and Unix environments. Last but not least, Wget supports background operation and is available freely under the GNU public license.

After downloading the pages and excluding pages that returned an HTTP 404 error response, the script computed the size of each downloaded site on the file system and updated the respective record in SQL Server with the calculated file size. Grouping the sites into categories was achieved by sorting the list of sites in ascending order and using 10,000 sites for each category.

3.3.2. System Profiling

Windows Performance Monitor (Perfmon), a tool that can be used to log performance data, was used to profile the system during experiments (Microsoft, 2010). In addition to having a command line interface, Perfmon includes a GUI that allows users to control the collection process. Perfmon was chosen because it's freely available, allows command line automation, is officially maintained by Microsoft, and produces data in a standard CSV format for analysis.

3.3.3. Thumbnailer Flow

The thumbnailer was implemented as a server application in Microsoft C#.NET. The system maintains a queue of web pages that are each individually processed. The process executes as follows:

1. The system spawns the specified number of worker threads and waits for new jobs.
2. A web page queue is populated, then the queue is locked and the first web page address is passed to one of the free worker threads for processing.
3. The thread spawns an external process that handles rendering the web page in a hidden Web browser and capturing the web page coordinates to a bitmap while including a one second delay to allow Flash, Java, and Javascript content to load.

4. The thread crops and resizes the bitmap to the desired thumbnail width of 220px while maintaining the vertical aspect ratio.
5. The thread saves the resulting thumbnail to the file system for later caching and future access.
6. The thread will mark itself as free, and the process begins again with step 2.

Based on the author's experience, a linear single threaded thumbnailer takes a minimum of a few seconds to process a single web page. The bulk of the time is wasted on loading the web page and providing an adequate delay to ensure that the full page loads. Because web page loading time cannot be controlled, the goal was to utilize multiple application threads to process more than one web page at a time. Processing a page includes loading it in a hidden browser, capturing and processing the thumbnail, and writing the file to disk. This satisfies the scalability requirement by ensuring that more web pages can be processed as additional hardware capacity is added.

3.4. Data Analysis

Scalability is presented in terms of the time it took to generate captures of the 10,000 web pages in the sample set for each size category. The system usage profiled was used to explain any anomalies in the data. The main goal of the application was the ability to increase web page throughput as the number of threads is increased, this means that the most important metric is the duration it takes for the thumbnails to be generated for the sample sets for a specific number of threads. In addition to the individual trial results, the processing duration is presented for each size category and thread count.

3.5. Summary

This section outlined what data is important in testing the system, how to collect this data, the various variables, and the operating environment where the system was tested. Scalability and reliability are naturally ambiguous terms; the hypotheses were defined to disambiguate the term *scalability*. The software's architecture was outlined and the testing methodology was presented. The next chapter will examine the experiment results.

CHAPTER 4. RESULTS

This chapter will discuss the timed results of the web page thumbnailer built by the author. The data presented compares the time it took to generate thumbnails of 10000 web pages with a varying number of capture threads and file sizes. The system was also benchmarked for CPU, memory, disk, and network usage throughout the trials.

The experiments were conducted using the system described in section 3.1. After the experiment completed, the profiling data was logged along with the thread count and size group. To analyze the benchmark data, the profiler's CSV logs were imported into Microsoft Excel for easier analysis.

The data is not perfect due to the uncontrollable nature of the Web. web pages have different response times that vary between requests. A residential cable internet connection was used, thus the speeds can heavily vary throughout the day. Additionally, the experiments were not run in a full production environment, so the system may have had processes running in the background that were unrelated to this research. To minimize these factors, the trials were conducted 10-12 times for each thread count and size category.

4.1. Sample Set Size Collection

Table 4.1 shows the web page size categorization collected from the set of 30,000 web pages collected from the Open Directory Project.

Table 4.1 Size Category Ranges

Size Category	Size Range (Kilobytes)	Mean	Median
Small	0.32 KB – 72 KB	31 KB	30 KB
Medium	72 KB - 272 KB	148 KB	137 KB
Large	272 KB – 39000 KB	765 KB	493 KB

Figures 4.1 and 4.2 are examples of small web pages. Figure 4.1 shows that the page only includes a few lines of text which explains the extremely small file size. Similarly, the second example in figure 4.2 shows a web page with one image and a limited amount of text.

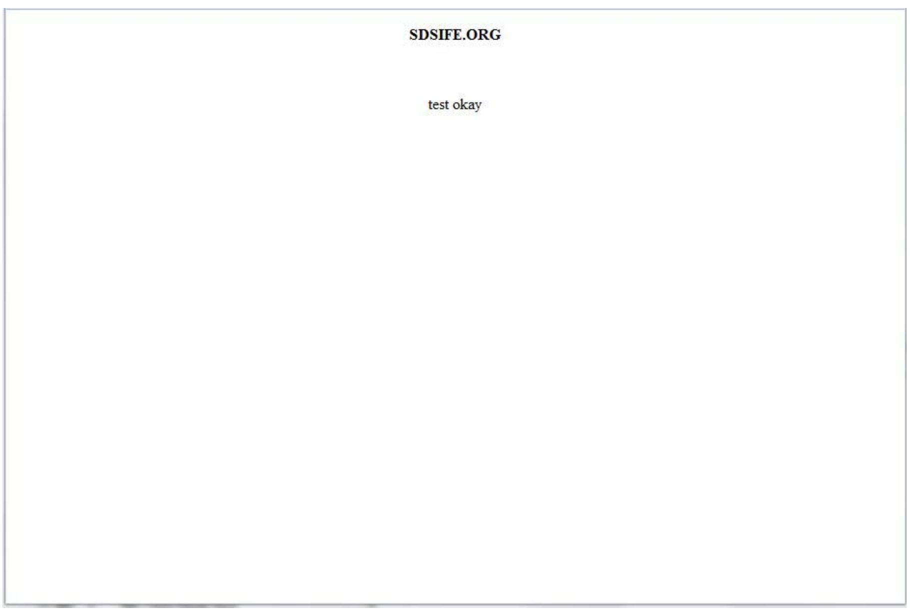

Figure 4.1 www.sdsife.org – 0.34 KB

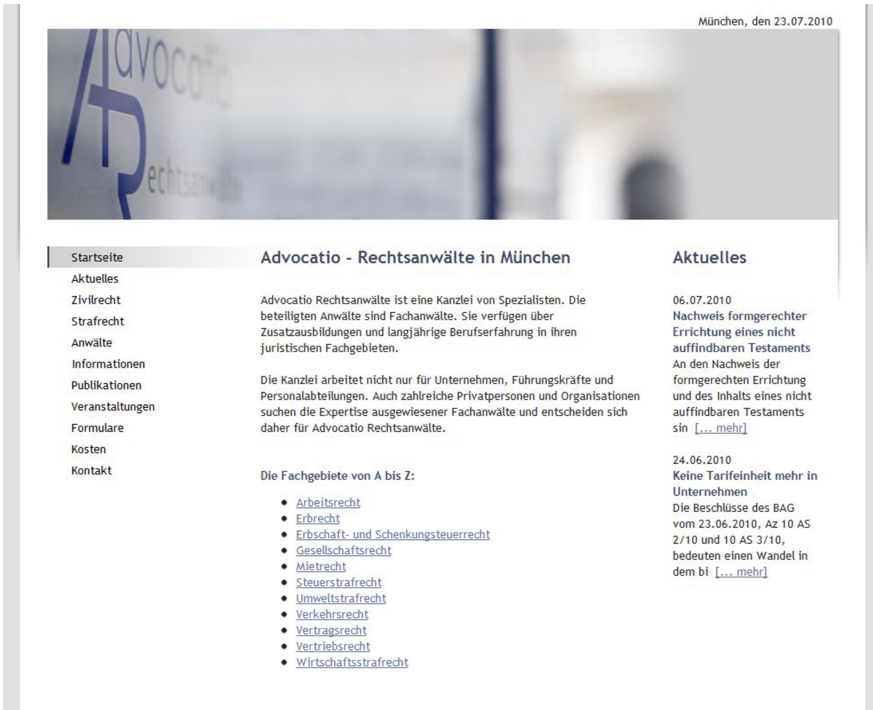

Figure 4.2 http://www.bornewasser-hacker.de/ – 33.8 KB

Examples of medium sized web pages are shown in figures 4.3 and 4.4.

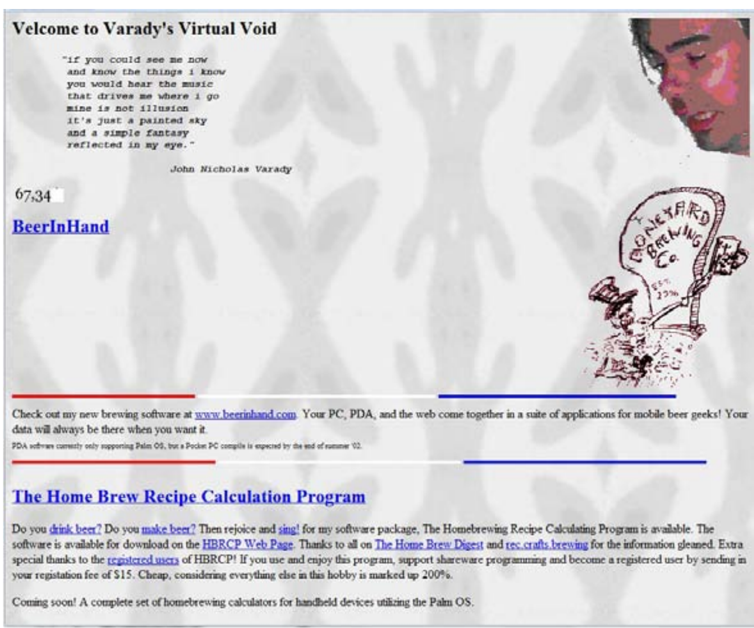

Figure 4.3 http://www.uberbeergeek.com/bih/ – 99.7 KB

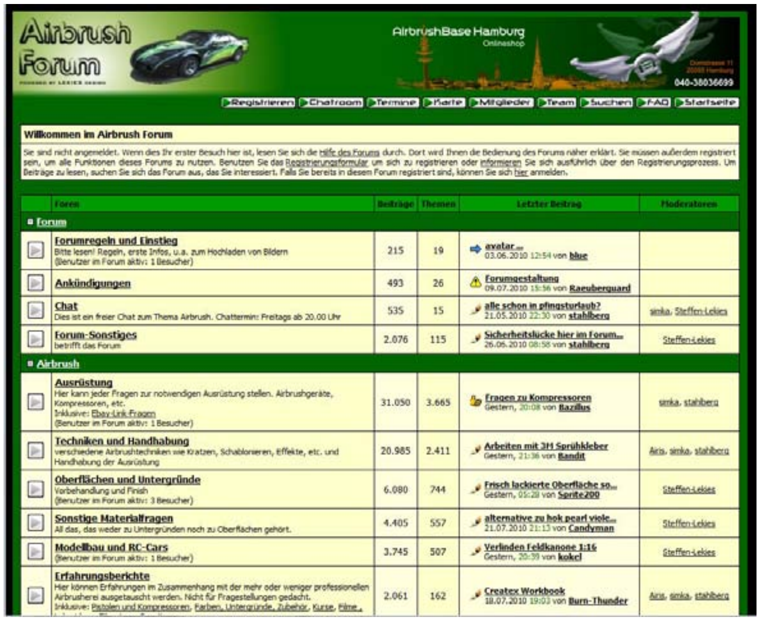

Figure 4.4 http://airbrush-forum.net/ – 167 KB

The next two figures illustrate examples of large web pages. What's not visible in Figure 4.5 is that many of the graphics on the page were animated GIFs which may have contributed to the large file size. In addition to the main content, the page in Figure 4.6 consists of an image scroller with approximately 15 images that load when the page is requested.

Figure 4.5 http://www.pinetree-golf-club.jp/ – 20.3 MB

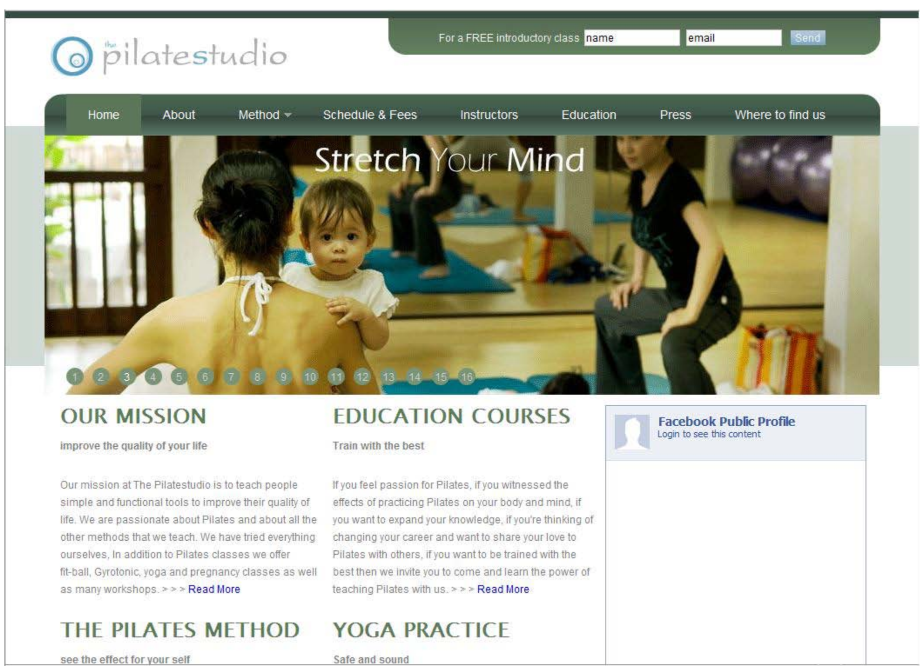

Figure 4.6 www.pilatesbangkok.com – 2.05 MB

4.2. Experiment Results

After completing the sample set analysis, a simple interface was created on the thumbnailer that allowed a user to input the number of web pages, threads, and select a size category. When all worker threads finished processing the queue of pages, the experiment data was logged. This data consisted of debugging messages and experiment variables such as duration, number of web pages, and the selected size category. The following figure shows a screenshot of the interface.

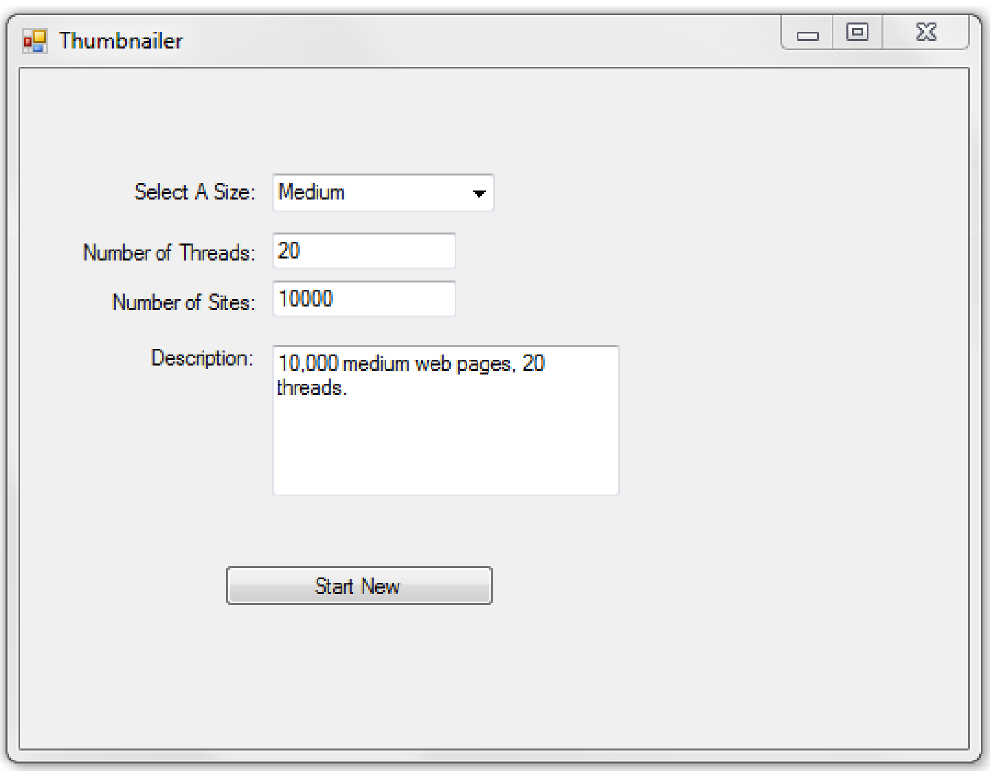

Figure 4.7 Thumbnailer Screenshot

The experiment was also conducted with a single thread to establish a baseline for the experiments. A single thread on large web pages can take a long amount of time to capture 10,000 pages. Instead of capturing the full 10,000 pages, the thumbnailer ran until it captured 6511 pages and the duration was logged. Table 4.2 shows the duration it took to capture 10,000 small and medium web pages, and 6511 large web pages with a single thread. Tables 4.3 and 4.4 show the durations in seconds it took to process the 10,000 web pages for each thread count and size group.

Table 4.2 Duration Baseline – Single Thread

	Small	Medium	Large
Number of Pages	10,000	10,000	6511
Duration (seconds)	35,928	50,019	56,026
Avg. Seconds/Page	3.5928	5.0019	8.606

Table 4.3 Duration Data: 10-50 Threads

Trial	10 Threads			20 Threads			30 Threads			50 Threads		
	Sm.	Med.	La.	Sm.	Med.	La.	Sm.	Med.	La.	Sm.	Med.	La.
1	4612.82	5960.86	11892.68	3420.60	5279.99	10463.02	3222.91	4995.71	8352.668	2866.93	4249.62	5513.35
2	4568.00	5761.86	11667.14	3932.50	5331.87	10314.57	3318.13	5032.60	8168.529	2903.13	4294.30	5520.86
3	4637.71	5971.74	12207.09	3821.61	4878.51	10378.94	3340.71	5105.40	8187.294	2896.99	4344.59	5515.26
4	4555.32	5677.30	12103.95	3470.06	5441.98	10153.25	3340.58	5101.27	8034.034	2985.71	4250.74	5512.79
5	4587.15	5755.81	12241.89	3573.02	5284.90	9962.55	3403.97	5062.52	7992.855	2872.73	4301.78	5601.51
6	4583.00	5777.21	11977.73	3666.34	5111.67	10068.87	3777.51	4989.65	8182.065	2934.22	4207.91	5589.91
7	4555.32	5804.72	12040.35	3544.11	5255.32	10272.85	3390.45	5046.77	8173.574	2933.56	4155.09	5593.13
8	4587.15	5843.18	11875.29	3723.24	4955.05	10226.14	3325.07	5208.73	8058.742	2904.74	4204.67	5545.58
9	4583.00	5827.68	11820.03	3645.75	4810.39	9898.62	3202.04	5226.06	8149.128	2946.84	4299.05	5548.98
10	4500.67	5855.83	11560.58	3637.04	4912.69	9917.35	3393.34	4809.58	7988.226	2834.62	4255.05	5522.92
Avg.	**4577.01**	**5823.62**	**11938.67**	**3643.43**	**5126.24**	**10165.62**	**3371.47**	**5057.83**	**8128.71**	**2907.95**	**4256.28**	**5546.43**

Table 4.4 Duration Data: 75-175 Threads

Trial	75 Threads			125 Threads			150 Threads			175 Threads		
	Sm.	Med.	La.	Sm.	Med.	La.	Sm.	Med.	La.	Sm.	Med.	La.
1	2410.92	3318.07	3826.76	1941.66	2185.28	--	1869.10	2092.84	3312.52	--	--	3219.28
2	2410.16	3264.78	3822.24	2064.93	2243.90	--	1836.04	2029.54	3373.91	--	--	3244.40
3	2441.98	3087.14	4056.65	2077.71	2106.93	--	1856.15	2045.20	3164.50	--	--	3301.82
4	2433.70	3079.59	3836.58	1936.03	2130.59	--	1841.27	2102.67	3414.76	--	--	3213.39
5	2472.65	3161.10	4002.97	1968.79	2256.66	--	1829.02	2130.05	3318.65	--	--	3232.05
6	2470.99	3238.67	3863.05	2092.78	2283.04	--	1864.53	2089.83	3326.03	--	--	3243.32
7	2422.07	3127.39	3943.23	2090.78	2209.83	--	1839.23	2038.59	3221.15	--	--	3295.19
8	2431.88	3286.71	3964.07	2041.89	2157.35	--	1826.80	1983.68	3211.05	--	--	3268.84
9	2411.65	3255.63	4087.72	1988.53	2173.25	--	1861.89	2003.58	3317.25	--	--	3243.88
10	2472.41	3340.20	3917.52	1960.29	2159.64	--	1845.35	2108.73	3331.32	--	--	3359.98
Avg.	**2437.84**	**3214.66**	**3932.08**	**2016.34**	**2189.98**	**--**	**1846.94**	**2062.47**	**3299.11**	**--**	**--**	**3261.94**

In figure 4.8, the duration was plotted against the thread count for small, medium, and large pages. Refer to section 3.2 for details on how trend lines were fitted.

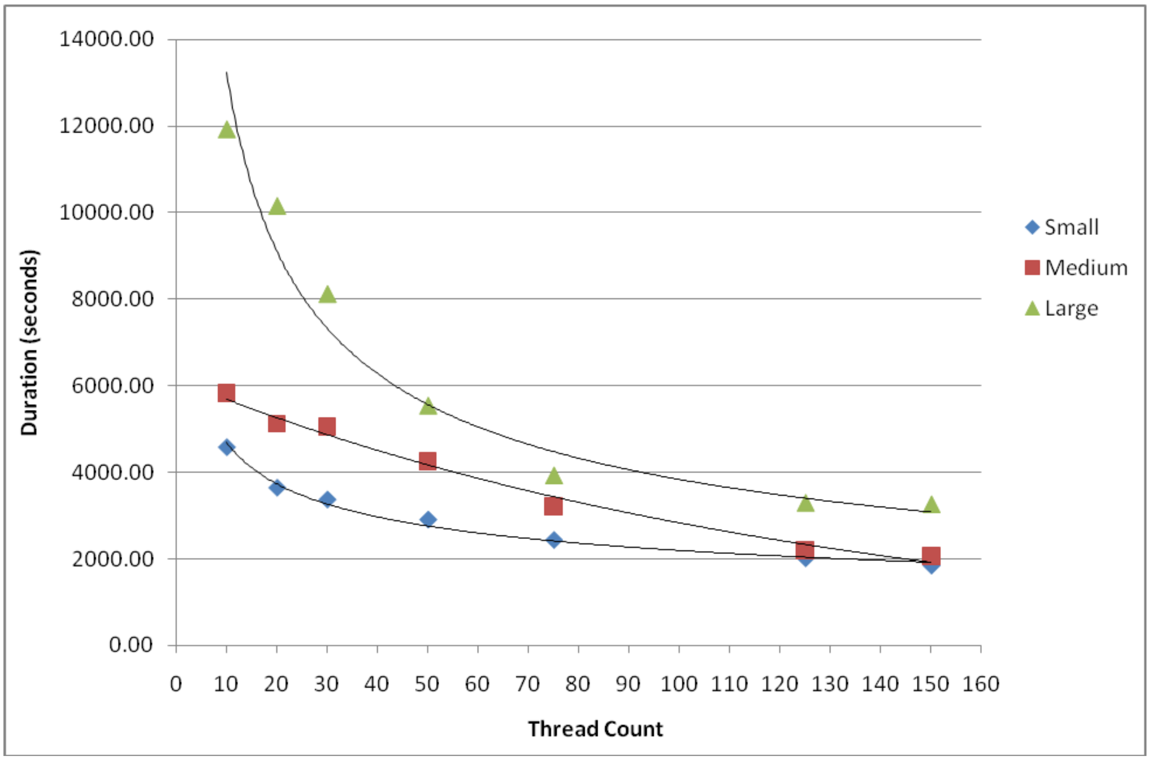

Figure 4.8 Duration vs. Thread Counts Chart

4.3. System Profile

As stated in the methodology chapter, Perfmon was used to benchmark the system. The appendix shows a full system profile for all data executions with accompanying field definitions.

4.4. Summary

This chapter presented the various results for each size group. Due to the complex nature of the Web, the durations varied slightly between trials. The next section will analyze these results.

CHAPTER 5. DATA ANALYSIS AND CONCLUSION

This chapter will examine the results presented in chapter 4 and provide insights based on the data collected. The system profile data will be discussed in relation to the duration data collected. Finally, thread count recommendations will be made and future research angles on this topic will be explored.

5.1. Findings

For small web pages, the thumbnailer's efficiency decreased as the thread count increased past 120. To examine the thumbnailer's efficiency, the slope was calculated between two adjacent points on the chart to observe whether or not an improvement occurs as the number of threads was increased. In this context, the thumbnailer's efficiency refers to the duration it took to process the web pages in the sample set; a higher efficiency indicates a lower number of seconds elapsed in processing the 10,000 web pages. Because the chart slopes downwards, a negative slope that approaches zero indicates that the thumbnailer was more efficient at processing the sample set, i.e. the 10,000 pages were processed in a smaller number of seconds. A zero slope indicates that an increased thread count does not provide any visible improvement in the thumbnailer's efficiency. Table 5.1 shows the average duration for each thread count in the small size category.

Table 5.1 Small Pages Thread Count Durations

Thread Count	Small Avg. Duration (seconds)
10	4577.01
20	3643.43
30	3371.47
50	2907.95
75	2437.84
125	2016.34
150	1846.94

To explain why the thumbnailer's efficiency decreased, the system profile can be examined. Table 5.2 below shows a partial system profile for small pages using 1, 125 and 150 threads.

Table 5.2 Small Pages Partial System Profile

Thread Count	Memory Pages Input/sec	Memory Pages/sec	Avg. Disk sec/Read	Avg. Disk sec/Write	Avg. Network Utilization (% 10 Mbit)
1	9	12	0.001	0.001	3.21 %
125	31	60	0.015	0.001	25.43 %
150	111	149	0.017	0.001	23.48 %

Based on the performance counters outlined in the research methodology, the profile shows that running the thumbnailer with 125 threads increased memory paging to disk. Paging occurs to resolve hard page faults which are caused by a process referring to a page that isn't currently in memory and must be retrieved from disk. As a result of paging, the disk read response time increased above the 15ms threshold recommended by Microsoft TechNet (2010b) when 125 or 150 threads were used. A 15ms read response time indicates that the disk is keeping up with demands with very little overhead to process more page faults.

Medium web pages exhibited similar results to small pages. The small and medium size categories had size differences of less than 200 KB which is not a major difference. Table 5.3 shows the average duration for each thread count in the medium size category.

Table 5.3 Medium Pages Thread Count Durations

Thread Count	Medium Avg. Duration (seconds)
10	5823.62
20	5126.24
30	5057.83
50	4256.28
75	3215.93
125	2190.65
150	2062.47

Table 5.4 shows the system profile for medium pages. The profile for 20 threads was also included to explain the abnormally small difference between 20 and 30 threads.

Table 5.4 Medium Pages Partial System Profile

Thread Count	Memory Pages Input/sec	Memory Pages/sec	Avg. Disk sec/Read	Max Disk sec/Read	Avg. Disk sec/Write	Avg. Network Utilization (% 10 Mbit)
1	7	7	0.001	0.339	0.001	3.61 %
20	9	11	0.003	0.522	0.001	26.47 %
30	144	145	0.004	2.547	0.001	25.35 %
125	38	93	0.007	0.328	0.001	28.23 %
150	129	129	0.009	0.566	0.001	25.14 %

The memory usage shows that page faults were high in the 125 and 150 threads. Additionally, 30 threads exhibited a very high paging rate that which explains the small improvement from when increasing the thread count from 20 to 30. Unfortunately, the non-production environment makes it difficult to diagnose the exact cause of the paging, but it was likely the result of an external Windows process that spawned throughout the 30 thread execution and consumed part of the system’s resources. Although the average read response time remained low, the 30 thread execution experienced a much higher maximum disk read latency. This suggests that the disk paging occurred in small bursts where the average response time was over 2 seconds rather than a consistently high response time.

The large size category tremendously benefited from an increased thread count, this is demonstrated by the sharp curve decline. For example, it took over 12,000 seconds to process the 10,000 pages using 10 threads, and a little over 3,000 seconds when using 150 threads. Adding more threads after the 150 thread mark did not lead to a further drop in the processing duration. Table 5.5 shows the average duration for each thread count in the large size category.

Table 5.5 Large Pages Thread Count Durations

Thread Count	Large Avg. Duration (seconds)
10	11938.67
20	10165.62
30	8128.71
50	5546.43
75	3932.08
150	3299.11
175	3261.94

Table 5.6 shows the system profile for large web pages. It appears that large web pages consumed the most network bandwidth. Memory paging increased when 150 threads or greater were used. This led to an increase in the disk response time past the 15ms threshold which led to a very slow response time. Additionally, small and medium pages showed a higher memory paging rate than large pages. This may have been by a larger amount of web pages that responded with an HTTP 404 error in the small and medium size categories. Although these pages were initially removed in the initial size collection, the experiments occurred over a period of 5 weeks which may have led to a larger number of 404s throughout the study.

Table 5.6 Large Pages Partial System Profile

Thread Count	Memory Pages Input/sec	Memory Pages/sec	Avg. Disk sec/Read	Avg. Disk sec/Write	Avg. Network Utilization (% 10 Mbit)
1	5	6	0.001	0.001	4.34 %
150	66	67	0.012	0.004	32.80 %
175	162	178	0.024	0.006	38.29 %

5.2. Experimental Evaluation

H_0: *As the number of simultaneous capture threads increases for a fixed number of web pages, the thumbnailer's process duration does not decrease.*

H_a: *As the number of simultaneous capture threads increases for a fixed number of web pages, the thumbnailer's process duration decreases.*

To test the alternative hypothesis, it must be shown that the thumbnailer processed the sample set of web pages in a smaller duration as the capture threads increased for each of the size categories. This was proven to be true in Figure 4.8, as the number of threads increased, the duration of processing the set of 10,000 web pages decreased. The slope calculations in section 5.1 show that it decreases as the thread count is increased. A zero slope indicates that an increase in thread count provides no visible improvement to the duration.

The second part in proving the alternative hypothesis is to examine the system profile and test that the thumbnailer cannot process more web pages when a certain hardware limit is approached. As shown in Tables 5.2, 5.4, and 5.6, the highest thread count experienced hardware issues that may have contributed to the limit approached. Small, medium, and large sites at different thread thresholds suffered from excessive memory paging to disk creating a disk bottleneck.

5.3. Recommendations

For all size groups, the main constraint was the disk response time when increasing thread counts. This may have been caused by the thumbnailer saving numerous screenshots to disk while other web pages were being downloaded. To achieve higher thread counts, the disk problems would have to be addressed by either optimizing the thumbnailer or adding multiple disks that can process the data. Large pages consumed the most network bandwidth, which means that they would require a faster connection to increase throughput. Although CPU usage was not an issue, small and medium pages used slightly more CPU than large web pages. This can be attributed to more thread switching caused by spawning additional threads in a shorter period of time as sites were captured (Narten, 1997). Specific thread count recommendations can be provided, but will most likely be different if any part of the test system changes. On the system used in this research, the thumbnailer showed marginal improvements in the 125-175 thread range for all size groups.

5.4. Conclusion & Future Research

Improving usability on the Web can benefit countless people in the world. Thumbnails have been shown to improve interfaces and increase user interaction. In this research, a scalable thumbnailer was built and a testing methodology was presented. The testing methodology can be re-used in any size-sensitive research. General recommendations were provided on the optimal thread counts for different web page sizes.

Future research can tackle other issues relating to web page thumbnails. Updating the existing thumbnails still remains a challenge; research can be conducted in checking the update frequency of web pages. A more granular size categorization might produce better thread count recommendations that may be more representative of web pages on the World Wide Web. The current thumbnailer can be improved from a performance standpoint by deferring disk access and optimizing the code. Finally, implementing a DNS pre-fetching module would speed up the process of loading web pages and improve the speed at which thumbnails can be generated.

5.5. Summary

This chapter analyzed the data for each size category and presented thread count recommendations. The conclusions were given and future angles on the research were presented.

BIBLIOGRAPHY

BIBLIOGRAPHY

Anderson, D., Cobb, J., Korpela, E., Lebofsky, M., & Werthimer, D. (2002). SETI@home: An Experiment in Public-Resource Computing. *Communications of the ACM*, *45*(11), 56-61.

Boldi, P., Codenotti, B., Santini, M., & Vigna, S. (2004). UbiCrawler: a scalable fully distributed Web crawler. *Software: Practice and Experience*, *34*(8).

Bondi, A. B. (2000). Characteristics of Scalability and Their Impact on Performance. *Proceedings of the 2nd international workshop on Software and performance.*

Broder, A. Z., Najork, M., & Wiener, J. L. (2003). Efficient URL caching for world wide Web crawling. *Proceedings of the twelfth international conference on World Wide Web - WWW '03*, 679. New York, New York, USA: ACM Press.

Chen, Y., Ma, W., & Zhang, H. (2003). Detecting web page structure for adaptive viewing on small form factor devices. *WWW2003*. Retrieved from http://www2003.org/cdrom/papers/refereed/p297/p297-chen.htm.

Clemson University. (n.d.). Linear Regression and Excel. *Physics Laboratory*. from http://phoenix.phys.clemson.edu/tutorials/excel/regression.html.

Cockburn, A., Greenberg, S., Jones, S., McKenzie, B., & Moyle, M. (2003). Improving Web Page Revisitation: Analysis, Design, and Evaluation. *Growth (Lakeland)*, 4, 6-7, 11-16.

benchmark. (n.d.). In *Computer Definitions*. Retrieved May 10th, 2010, from http://www.yourdictionary.com/computer/benchmark

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S., et al. (2002). Unraveling the Web Services Web. *IEEE Internet Computing*, (April).

Czerwinski, M. P., Dantzich, M. V., Robertson, G., Hoffman, H. (1998). The Contribution of Thumbnail Image, Mouse-over Text and Spatial Location Memory to Web Page Retrieval in 3D. *Microsoft Research.*

reliable. (n.d.). *Dictionary.com Unabridged*. Retrieved February 25, 2010, from Dictionary.com Website: http://dictionary.reference.com/browse/reliable

Foulonneau, M., Habing, T., & Cole, T. (2006). Automated Capture of Thumbnails and Thumbshots for Use by Metadata Aggregation Services. *D-Lib Magazine*. Urbana-Champaign. Retrieved from http://www.dlib.org/dlib/january06/foulonneau/01foulonneau.html.

Free Software Foundation. (2003). Introduction to GNU Wget. Retrieved from http://www.gnu.org/software/wget/.

Hightower, R. R., Ring, L. T., Helfman, J. I., Bederson, B. B., & Hollan, J. D. (1998). Graphical Multiscale Web Histories: A Study of PadPrints. *Cognitive Science*.

Internet World Stats (September 2009). *Internet Usage Statistics* Retrieved March 10, 2010,
from http://www.internetworldstats.com/stats.htm

Kaasten, S., Greenberg, S., & Edwards, C. (2001). How People Recognize Previously Seen web pages from Titles, URLs and Thumbnails. *Proceedings of Human Computer Interaction 2002*.

Kaasten, S., & Greenberg, S. (2001). Integrating back, history and bookmarks in Web browsers. *CHI '01 extended abstracts on Human factors in computing systems - CHI '01*, 379. New York, New York, USA

Microsoft. (2010). Windows Performance Monitor. *Technet*. Retrieved from http://technet.microsoft.com/en-us/library/cc749249.aspx.

MSDN. (2004). Chapter 15 Measuring .NET Application Performance. *Prescriptive Architecture*. Retrieved from http://msdn.microsoft.com/en-us/library/ms998579.aspx.

MSDN. (n.d.). DA0505: Average Private Bytes allocated for the Process being profiled. Retrieved from http://msdn.microsoft.com/en-us/library/ee815699.aspx.

MSDN. (2010a). NEWID. *Transact-SQL Reference*. Retrieved from http://msdn.microsoft.com/en-us/library/aa276822(SQL.80).aspx

MSDN. (2010b). Scheduling Priorities. Retrieved from http://msdn.microsoft.com/en-us/library/ms685100(VS.85).aspx.

Narten, T. (1997). Mechanics of Thread Switching. Nachos Threads. Retrieved from http://www.cs.duke.edu/~narten/110/nachos/main/node13.html.

Nationmaster.com. (n.d.). Encyclopedia > Lossless data compression. Retrieved from http://www.statemaster.com/encyclopedia/Lossless-data-compression.

Netscape. (2010). ODP - Open Directory Project. Retrieved from http://www.dmoz.org/.

W3C RDF Group. (2004). RDF. Retrieved from http://www.w3.org/RDF/.

concurrent. (n.d.) *The American Heritage® Dictionary of the English Language, Fourth Edition*. (2003). Retrieved April 9 2010 from http://www.thefreedictionary.com/concurrent

Microsoft Technet. (2010a). Monitoring Multiple Processor Computers. *Microsoft TechNet Library*. Retrieved from: http://technet.microsoft.com/en-us/library/cc722478.aspx.

Microsoft Technet. (2010b). Performance data from Average Disk sec/Read counter of the PhysicalDisk performance object. *Microsoft Technet Library*. Retrieved from http://technet.microsoft.com/en-us/library/aa996220(EXCHG.80).aspx.

Microsoft Technet. (2005a). Ruling Out Disk-Bound Problems. *Microsoft TechNet Library*. Retrieved from: http://technet.microsoft.com/en-us/library/aa997558(EXCHG.65).aspx.

Microsoft Technet. (2005b). Ruling Out Memory-Bound Problems. *Microsoft TechNet Library*. Retrieved from: http://technet.microsoft.com/en-us/library/aa998384(EXCHG.65).aspx.

thread. (n.d.) *The American Heritage® Dictionary of the English Language, Fourth Edition*. (2003). Retrieved May 11 2010 from http://www.thefreedictionary.com/thread

Teevan, J., Cutrell, E., Fisher, D., Drucker, S. M., Ramos, G., Hu, C., et al. (2009). Visual Snippets: Summarizing web pages for Search and Revisitation. *Proceedings of the 27th international conference on Human factors in computing systems*.

World Wide Web Consortium (n.d.). Browser Statistics. Retrieved from http://www.w3schools.com/browsers/browsers_stats.asp.

World Wide Web Consortium, (W3C). (1999). Header Field Definitions. *Hypertext Transfer Protocol -- HTTP/1.1*. Retrieved from http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html.

World Wide Web Consortium (2003). Portable Network Graphics (PNG) Specification (Second Edition). Retrieved from http://www.w3.org/TR/PNG/.

Yoo, B., Lea, J., & Kim, Y. (2008). The Seamless Browser: Enhancing the Speed of Web Browsing by Zooming and Preview Thumbnails. *Proceeding of the 17th international conference on World Wide Web - WWW '08*, 1019. New York, New York, USA: ACM Press.

APPENDIX

Appendix Table: Summarized System Profile Data

Size	Thread Count	Disk sec/Read		Disk sec/Write		Memory Pages/sec		Memory Pages Input/sec	Private Bytes (Memory Usage)			Processor Queue Length	% Total Processor Time (Thumbnailer process)		Network Utilization (bytes)	
		Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Min.	Avg.	Max	Min./Avg./Max.	Avg.	Max.	Avg.	Max.
Small	1	0.001	0.526	0.001	0.065	12	14240	9	16,228,352	143,672,185	291,876,864	0/0/4	1.75	38.5	42,106	793277
	10	0.002	1.842	0.001	0.098	13	4624	13	37,068,800	249,291,686	416,649,216	0/0/15	15.75	49.5	274953	708614
	20	0.008	0.507	0.001	0.072	50	5138	46	23,945,216	256,515,674	406,814,720	0/1/16	25.25	72.75	336,237	708,500
	30	0.004	0.717	0.001	0.074	56	11579	46	37,052,416	362,701,691	471,797,760	0/2/21	22.5	66.25	293,681	583,920
	50	0.003	1.277	0.001	0.093	27	4550	23	37,068,800	465,134,902	577,126,400	0/1/23	26.75	66	335,510	627,941
	75	0.018	0.562	0.002	0.213	191	19440	191	37,064,704	584,137,784	677,998,592	0/3/73	29.5	59	351,662	642,228
	125	0.015	0.239	0.001	0.019	60	1518	31	37,064,704	827,205,106	931,864,576	0/3/23	33.5	65.25	333,354	540,329
	150	0.017	0.455	0.001	0.096	149	6364	111	37,060,608	932,964,116	1,039,273,984	0/2/24	27.5	66.75	307,804	522,897
Medium	1	0.001	0.339	0.001	0.116	7	4966	7	16,220,160	152,283,277	233,365,504	0/0/5	2.25	32.25	47,362	932,926
	10	0.002	1.291	0.001	0.059	26	9121	26	37,072,896	294,269,837	549,629,952	0/1/18	16.75	51.75	316,429	817,192
	20	0.003	0.522	0.001	0.071	11	4125	9	37,048,320	339,412,776	600,248,320	0/1/15	20.5	65.75	346,970	644,678
	30	0.004	2.547	0.001	0.103	145	22206	144	37,052,416	397,514,188	507,994,112	0/1/15	20.75	58.5	332,316	659,895
	50	0.007	0.43	0.001	0.121	104	22410	77	37,068,800	514,213,244	620,486,656	0/2/26	24.75	64	395,154	739,406
	75	0.01	0.522	0.001	0.077	26	5559	26	37,064,704	629,864,484	715,038,720	0/0/12	18.25	57.75	339,667	644,682
	125	0.007	0.328	0.001	0.089	93	8772	38	37,089,280	874,949,179	964,198,400	0/2/35	27.5	61.25	370,054	611,355
	150	0.009	0.566	0.001	0.119	129	20253	129	37,081,088	980,178,937	1,050,140,672	0/1/145	20.75	88.5	329,536	581,174
Large	1	0.001	0.495	0.001	0.086	6	11387	5	37,060,124	289,201,155	556,951,102	0/0/4	3.14	49.65	56,940	894,441
	10	0.003	1.036	0.001	0.067	76	18726	76	37,060,608	336,201,265	667,234,304	0/0/19	13.5	55	451,918	937,790
	20	0.002	1.316	0.001	0.148	97	21868	97	37,068,800	388,529,655	580,481,024	0/0/13	10	36.25	447,538	853,519
	30	0.002	0.78	0.001	0.129	8	3576	8	37,085,184	459,200,284	723,447,808	0/1/17	17.25	61.25	477,341	869,073
	50	0.01	1.997	0.001	0.063	96	21175	95	37,117,952	558,826,926	756,109,312	0/1/19	19.75	59.75	485,556	917,988

	75	0.01	1.89	0.001	0.056	18	4330	17	37,068,800	687,346,633	807,063,552	0/2/24	22.5	66.75	488,265	808,648
	125	0.009	0.241	0.001	0.038	10	3358	10	37,081,088	932,572,628	1,055,809,536	0/1/41	17	70.5	432,184	693,967
	150	0.012	0.547	0.004	0.099	67	20332	66	37,072,896	1,021,357,689	1,105,801,216	0/1/147	23.25	78.5	429,859	701,404
	175	0.024	1.231	0.006	0.114	178	23159	162	37,192,842	1,182,331,791	1,236,755,228	0/2/162	26	93.75	501,913	802,755

Appendix Definitions

1. Avg. Disk sec/read: Indicates the average time it takes to read data from the disk in seconds. According to Microsoft, the average value should be below 10 ms with spikes not being higher than 50 ms.
2. Avg. Disk sec/write: Indicates the average time it takes to write data from the disk in seconds. The average value should be below 10 ms and spikes should be no higher than 50 ms (Microsoft TechNet, 2005a).
3. Memory pages/sec: Indicates the rate at which pages are read from or written to disk. Microsoft TechNet (2005b) states that this value should be below 1,000 at all times.
4. Memory Pages Input/sec: The rate at which pages are read from the disk to resolve hard page faults.
5. Private Bytes: The maximum amount of virtual memory that the thumbnailer process has currently allocated in bytes. Private bytes are virtual memory locations that can only be accessed by the threads running inside the allocating process (MSDN, n.d.).
6. Processor Queue Length (PQL): The number of threads in the processor queue. The significance of the PQL is that if there are more tasks to run than the processors can handle, threads queue up. A sustained queue of more than two threads indicates a processor bottleneck (MSDN, 2004).
7. % Processor Time: The sum of processor time on each processor for all threads of the thumbnailer process (Microsoft TechNet, 2010a). This value is divided by the number of processors to produce an average.
8. Network Utilization in Megabits: Measured by collecting the total amount of bytes transferred over the network adapter and converting it to megabits.