Scholar Search

New Mexico

Supercomputing Challenge

Final Report

March 28, 2013

Team 87

Quemado High School

--Team Members--

Justin Miller

Sam Farr

Stefan Beauchamp

Sam Eberle

Execurive Summary

Schools nationwide spend millions of dollars to keep students safe online. We investigated the principles of large scale search engines and high demand web filtering software. The primary goal of the project was to define exactly what a search engine does and how they work. Knowing that building a full scale search engine that has the ability to index the internet at a high pace was not feasible, we decided to narrow the scope of the question down to building a small scale search engine. This search engine would have been able to filter out known inappropriate material such as proxy servers, malicious websites, pornography, and websites that encourage drugs, violence, and alcohol abuse. We believed that if we were successful at building such a search engine, we would be able to further expand on the project and design the crawlers to handle larger loads as well and be able to classify content more accurately.

To accomplish our goal, we used several different computer programming languages. First, we wanted to be able to monitor the search engines index process in real-time. The platform we designed to do this was programmed in PHP. Next, a more critical aspect of the project, the crawler, was programmed (also in PHP). The use of PHP was questioned but we chose this language because it has the ability to perform tasks at a fast pace, it is stable, secured, and can handle large loads. To filter the index, we have another PHP platform that monitors the pages that are being indexed at the time for keywords and known bad phrases.

Our final goal was to have the search engine create a user friendly computational display of its progress and current index side. At the time of the deadline, we do have a web based progress viewing platform but we hope to make it more user friendly and graphically appealing to the eye.

Defined Problem

The primary goal of this project is to investigate how search engines work and successfully build a small scale search engine that filters search results and displays only information that is appropriate for students.

Solution Path

Time constraints limited the search engine to be small scale, so performance was not a critical variable. It was believed that because it was small won't have a huge load, a less complicated programming language that is capable of executing commands in the simplest manner possible would be best beneficial to investigating the data being processed.

To start off, it was decided that PHP was going to be the primary programming language. HTML and JavaScript can be used within a PHP document so the need for third-party plug-ins was not necessary. PHP could also directly communicate with multiple database servers. We chose MySQL Server because it was designed around Windows. Windows Server 2008 and 2012 were the choice operating systems because of the simplicity of the user interface and the entire team was very familiar with Windows.

There are many PHP website crawlers that are downloadable from the internet; however none of the open source crawlers are designed to index multiple sources at one time, nor are they designed to allow users to install the crawler on multiple computers and index in parallel. To ensure the best performance, seven crawler computers were setup with Windows Server 2008. These computers had the PHP crawler installed and also had a direct connection to the external hard drive that held the database. Two more computers were setup with Windows Server 2008, but had DNS roles installed. These ran the DNS features for the website that would operate the search

engine. One computer with Windows Server 2012 was setup; this computer had multiple roles. First, it ran the internet information services. In addition, it was a domain controller for the cluster network. This computer also had two network cards installed; one of which had a dedicated IP setup so that the internet information service roles could work properly, and the other had a direct connection to the cluster network.

The cluster network was made up of one modem and one 8 port 10/100mb switch. The modem was purchased online and was not supplied by the internet service provider. It was configured to a dedicated IP and the DNS settings were set to direct the two DNS servers that were setup for the search engine. The modem has 7 gigabit ports which were not enough. There were a total of 9 devices connected to the cluster network: 7 crawlers, the network hard drive, and the primary domain controller. Since there were not enough ports in the modem itself, the switch was installed. All the computers were connected via Ethernet cable. This setup made up the entire network that ran the search engine.

The PHP crawler looks for the parent domain and uses it as the primary target. It follows all other targets found in the parent domain. To tell the crawler what domain(s) to use, the user must edit another PHP file that contains a list of seeds. The crawler code looks like this:

```
<?php
	set_time_limit (0);
	$include_dir = "../include";
	include "auth.php";
	require_once ("$include_dir/commonfuncs.php");
	$all = 0;
	extract (getHttpVars());
	$settings_dir =  "../settings";
	require_once ("$settings_dir/conf.php");
	include "messages.php";
	include "spiderfuncs.php";
	error_reporting (E_ALL ^ E_NOTICE ^ E_WARNING);
	$delay_time = 0;
	$command_line = 0;
	if (isset($_SERVER['argv']) && $_SERVER['argc'] >= 2) {
		$command_line = 1;
		$ac = 1; //argument counter
		while ($ac < (count($_SERVER['argv']))) {
```

```
                $arg = $_SERVER['argv'][$ac];
                if ($arg  == '-all') {
                        $all = 1;
                        break;
                } else if ($arg  == '-u') {
                        $url = $_SERVER['argv'][$ac+1];
                        $ac= $ac+2;
                } else if ($arg  == '-f') {
                        $soption = 'full';
                        $ac++;
                } else if ($arg == '-d') {
                        $soption = 'level';
                        $maxlevel =  $_SERVER['argv'][$ac+1];;
                        $ac= $ac+2;
                } else if ($arg == '-l') {
                        $domaincb = 1;
                        $ac++;
                } else if ($arg == '-r') {
                        $reindex = 1;
                        $ac++;
                } else if ($arg  == '-m') {
                        $in =  str_replace("\ \ n", chr(10), $_SERVER['argv'][$ac+1]);
                        $ac= $ac+2;
                } else if ($arg  == '-n') {
                        $out =  str_replace("\ \ n", chr(10), $_SERVER['argv'][$ac+1]);
                        $ac= $ac+2;
                } else {
                        commandline_help();
                        die();
                }

        }
}
if (isset($soption) && $soption == 'full') {
        $maxlevel = -1;

}

if (!isset($domaincb)) {
        $domaincb = 0;
}
if(!isset($reindex)) {
        $reindex=0;
}

if(!isset($maxlevel)) {
        $maxlevel=0;
```

```
	}

	if ($keep_log) {
		if ($log_format=="html") {
			$log_file =  $log_dir."/".Date("ymdHi").".html";
		} else {
			$log_file =  $log_dir."/".Date("ymdHi").".log";
		}

		if (!$log_handle = fopen($log_file, 'w')) {
			die ("Logging option is set, but cannot open file for logging.");
		}
	}

	if ($all ==  1) {
		index_all();
	} else {

		if ($reindex == 1 && $command_line == 1) {
			$result=mysql_query("select url, spider_depth, required, disallowed,
can_leave_domain from ".$mysql_table_prefix."sites where url='$url'");
			echo mysql_error();
			if($row=mysql_fetch_row($result)) {
				$url = $row[0];
				$maxlevel = $row[1];
				$in= $row[2];
				$out = $row[3];
				$domaincb = $row[4];
				if ($domaincb=='') {
					$domaincb=0;
				}
				if ($maxlevel == -1) {
					$soption = 'full';
				} else {
					$soption = 'level';
				}
			}

		}
		if (!isset($in)) {
			$in = "";
		}
		if (!isset($out)) {
			$out = "";
		}

		index_site($url, $reindex, $maxlevel, $soption, $in, $out, $domaincb);
```

```
	}

	$tmp_urls  = Array();

	function microtime_float(){
	  list($usec, $sec) = explode(" ", microtime());
	  return ((float)$usec + (float)$sec);
	}

	function index_url($url, $level, $site_id, $md5sum, $domain, $indexdate, $sessid,
$can_leave_domain, $reindex) {
		global $entities, $min_delay;
		global $command_line;
		global $min_words_per_page;
		global $supdomain;
		global $mysql_table_prefix, $user_agent, $tmp_urls, $delay_time,
$domain_arr;
		$needsReindex = 1;
		$deletable = 0;

		$url_status = url_status($url);
		$thislevel = $level - 1;

		if (strstr($url_status['state'], "Relocation")) {
			$url = preg_replace("/ /", "", url_purify($url_status['path'], $url,
$can_leave_domain));

			if ($url <> '') {
				$result = mysql_query("select link from
".$mysql_table_prefix."temp where link='$url' && id = '$sessid'");
				echo mysql_error();
				$rows = mysql_numrows($result);
				if ($rows == 0) {
					mysql_query ("insert into ".$mysql_table_prefix."temp
(link, level, id) values ('$url', '$level', '$sessid')");
					echo mysql_error();
				}
			}

			$url_status['state'] == "redirected";
		}

		/*
		if ($indexdate <> '' && $url_status['date'] <> '') {
			if ($indexdate > $url_status['date']) {
```

```
                        $url_status['state'] = "Date checked. Page contents not
changed";
                        $needsReindex = 0;
                }
        }*/
        ini_set("user_agent", $user_agent);
        if ($url_status['state'] == 'ok') {
                $OKtoIndex = 1;
                $file_read_error = 0;

                if (time() - $delay_time < $min_delay) {
                        sleep ($min_delay- (time() - $delay_time));
                }
                $delay_time = time();
                if (!fst_lt_snd(phpversion(), "4.3.0")) {
                        $file = file_get_contents($url);
                        if ($file === FALSE) {
                                $file_read_error = 1;
                        }
                } else {
                        $fl = @fopen($url, "r");
                        if ($fl) {
                                while ($buffer = @fgets($fl, 4096)) {
                                        $file .= $buffer;
                                }
                        } else {
                                $file_read_error = 1;
                        }

                        fclose ($fl);
                }
                if ($file_read_error) {
                        $contents = getFileContents($url);
                        $file = $contents['file'];
                }


                $pageSize = number_format(strlen($file)/1024, 2, ".", "");
                printPageSizeReport($pageSize);

                if ($url_status['content'] != 'text') {
                        $file = extract_text($file, $url_status['content']);
                }

                printStandardReport('starting', $command_line);
```

```
                $newmd5sum = md5($file);


                if ($md5sum == $newmd5sum) {
                        printStandardReport('md5notChanged',$command_line);
                        $OKtoIndex = 0;
                } else if (isDuplicateMD5($newmd5sum)) {
                        $OKtoIndex = 0;
                        printStandardReport('duplicate',$command_line);
                }

                if (($md5sum != $newmd5sum || $reindex ==1) && $OKtoIndex ==
1) {
                        $urlparts = parse_url($url);
                        $newdomain = $urlparts['host'];
                        $type = 0;

        /*              if ($newdomain <> $domain)
                                $domainChanged = 1;

                        if ($domaincb==1) {
                                $start = strlen($newdomain) - strlen($supdomain);
                                if (substr($newdomain, $start) == $supdomain) {
                                        $domainChanged = 0;
                                }
                        }*/

                        // remove link to css file
                        //get all links from file
                        $data = clean_file($file, $url, $url_status['content']);

                        if ($data['noindex'] == 1) {
                                $OKtoIndex = 0;
                                $deletable = 1;
                                printStandardReport('metaNoindex',$command_line);
                        }


                        $wordarray = unique_array(explode(" ", $data['content']));

                        if ($data['nofollow'] != 1) {
                                $links = get_links($file, $url, $can_leave_domain,
$data['base']);
                                $links = distinct_array($links);
                                $all_links = count($links);
                                $numoflinks = 0;
```

```
                                        //if there are any, add to the temp table, but only if
there isnt such url already
                                        if (is_array($links)) {
                                                reset ($links);

                                                while ($thislink = each($links)) {
                                                        if ($tmp_urls[$thislink[1]] != 1) {
                                                                $tmp_urls[$thislink[1]] = 1;
                                                                $numoflinks++;
                                                                mysql_query ("insert into
".$mysql_table_prefix."temp (link, level, id) values ('$thislink[1]', '$level', '$sessid')");
                                                                echo mysql_error();
                                                        }
                                                }
                                        }
                                } else {
                                        printStandardReport('noFollow',$command_line);
                                }

                                if ($OKtoIndex == 1) {

                                        $title = $data['title'];
                                        $host = $data['host'];
                                        $path = $data['path'];
                                        $fulltxt = $data['fulltext'];
                                        $desc = substr($data['description'], 0,254);
                                        $url_parts = parse_url($url);
                                        $domain_for_db = $url_parts['host'];

                                        if (isset($domain_arr[$domain_for_db])) {
                                                $dom_id = $domain_arr[$domain_for_db];
                                        } else {
                                                mysql_query("insert into
".$mysql_table_prefix."domains (domain) values ('$domain_for_db')");
                                                $dom_id = mysql_insert_id();
                                                $domain_arr[$domain_for_db] = $dom_id;
                                        }

                                        $wordarray = calc_weights ($wordarray, $title, $host,
$path, $data['keywords']);

                                        //if there are words to index, add the link to the
database, get its id, and add the word + their relation
                                        if (is_array($wordarray) && count($wordarray) >
$min_words_per_page) {
                                                if ($md5sum == '') {
```

```
                                                mysql_query ("insert into
".$mysql_table_prefix."links (site_id, url, title, description, fulltxt, indexdate, size, md5sum,
level) values ('$site_id', '$url', '$title', '$desc', '$fulltxt', curdate(), '$pageSize',
'$newmd5sum', $thislevel)");
                                                echo mysql_error();
                                                $result = mysql_query("select link_id from
".$mysql_table_prefix."links where url='$url'");
                                                echo mysql_error();
                                                $row = mysql_fetch_row($result);
                                                $link_id = $row[0];

                                                save_keywords($wordarray, $link_id,
$dom_id);

                                                printStandardReport('indexed',
$command_line);
                                        }else if (($md5sum <> '') && ($md5sum <>
$newmd5sum)) { //if page has changed, start updating

                                                $result = mysql_query("select link_id from
".$mysql_table_prefix."links where url='$url'");
                                                echo mysql_error();
                                                $row = mysql_fetch_row($result);
                                                $link_id = $row[0];
                                                for ($i=0;$i<=15; $i++) {
                                                        $char = dechex($i);
                                                        mysql_query ("delete from
".$mysql_table_prefix."link_keyword$char where link_id=$link_id");
                                                        echo mysql_error();
                                                }
                                                save_keywords($wordarray, $link_id,
$dom_id);
                                                $query = "update
".$mysql_table_prefix."links set title='$title', description ='$desc', fulltxt = '$fulltxt',
indexdate=now(), size = '$pageSize', md5sum='$newmd5sum', level=$thislevel where
link_id=$link_id";
                                                mysql_query($query);
                                                echo mysql_error();
                                                printStandardReport('re-indexed',
$command_line);
                                        }
                                }else {
                                        printStandardReport('minWords',
$command_line);

                                }
                        }
```

```
                }
            } else {
                $deletable = 1;
                printUrlStatus($url_status['state'], $command_line);

            }
            if ($reindex ==1 && $deletable == 1) {
                check_for_removal($url);
            } else if ($reindex == 1) {

            }
            if (!isset($all_links)) {
                $all_links = 0;
            }
            if (!isset($numoflinks)) {
                $numoflinks = 0;
            }
            printLinksReport($numoflinks, $all_links, $command_line);
        }


        function index_site($url, $reindex, $maxlevel, $soption, $url_inc, $url_not_inc,
$can_leave_domain) {
            global $mysql_table_prefix, $command_line, $mainurl,  $tmp_urls,
$domain_arr, $all_keywords;
            if (!isset($all_keywords)) {
                $result = mysql_query("select keyword_ID, keyword from
".$mysql_table_prefix."keywords");
                echo mysql_error();
                while($row=mysql_fetch_array($result)) {
                    $all_keywords[addslashes($row[1])] = $row[0];
                }
            }
            $compurl = parse_url($url);
            if ($compurl['path'] == '')
                $url = $url . "/";

            $t = microtime();
            $a =  getenv("REMOTE_ADDR");
            $sessid = md5 ($t.$a);


            $urlparts = parse_url($url);

            $domain = $urlparts['host'];
            if (isset($urlparts['port'])) {
                $port = (int)$urlparts['port'];
```

```
		}else {
			$port = 80;
		}



		$result = mysql_query("select site_id from ".$mysql_table_prefix."sites where
url='$url'");
		echo mysql_error();
		$row = mysql_fetch_row($result);
		$site_id = $row[0];

		if ($site_id != "" && $reindex == 1) {
			mysql_query ("insert into ".$mysql_table_prefix."temp (link, level, id)
values ('$url', 0, '$sessid')");
			echo mysql_error();
			$result = mysql_query("select url, level from
".$mysql_table_prefix."links where site_id = $site_id");
			while ($row = mysql_fetch_array($result)) {
				$site_link = $row['url'];
				$link_level = $row['level'];
				if ($site_link != $url) {
					mysql_query ("insert into ".$mysql_table_prefix."temp
(link, level, id) values ('$site_link', $link_level, '$sessid')");
				}
			}

			$qry = "update ".$mysql_table_prefix."sites set indexdate=now(),
spider_depth = $maxlevel, required = '$url_inc'," .
					"disallowed = '$url_not_inc',
can_leave_domain=$can_leave_domain where site_id=$site_id";
			mysql_query ($qry);
			echo mysql_error();
		} else if ($site_id == '') {
			mysql_query ("insert into ".$mysql_table_prefix."sites (url, indexdate,
spider_depth, required, disallowed, can_leave_domain) " .
					"values ('$url', now(), $maxlevel, '$url_inc',
'$url_not_inc', $can_leave_domain)");
			echo mysql_error();
			$result = mysql_query("select site_ID from ".$mysql_table_prefix."sites
where url='$url'");
			$row = mysql_fetch_row($result);
			$site_id = $row[0];
		} else {
			mysql_query ("update ".$mysql_table_prefix."sites set
indexdate=now(), spider_depth = $maxlevel, required = '$url_inc'," .
```

```
                                        "disallowed = '$url_not_inc',
can_leave_domain=$can_leave_domain where site_id=$site_id");
                        echo mysql_error();
                }


                $result = mysql_query("select site_id, temp_id, level, count, num from
".$mysql_table_prefix."pending where site_id='$site_id'");
                echo mysql_error();
                $row = mysql_fetch_row($result);
                $pending = $row[0];
                $level = 0;
                $domain_arr = get_domains();
                if ($pending == '') {
                        mysql_query ("insert into ".$mysql_table_prefix."temp (link, level, id)
values ('$url', 0, '$sessid')");
                        echo mysql_error();
                } else if ($pending != '') {
                        printStandardReport('continueSuspended',$command_line);
                        mysql_query("select temp_id, level, count from
".$mysql_table_prefix."pending where site_id='$site_id'");
                        echo mysql_error();
                        $sessid = $row[1];
                        $level = $row[2];
                        $pend_count = $row[3] + 1;
                        $num = $row[4];
                        $pending = 1;
                        $tmp_urls = get_temp_urls($sessid);
                }

                if ($reindex != 1) {
                        mysql_query ("insert into ".$mysql_table_prefix."pending (site_id,
temp_id, level, count) values ('$site_id', '$sessid', '0', '0')");
                        echo mysql_error();
                }


                $time = time();


                $omit = check_robot_txt($url);

                printHeader ($omit, $url, $command_line);


                $mainurl = $url;
                $num = 0;
```

```
		while (($level <= $maxlevel && $soption == 'level') || ($soption == 'full')) {
			if ($pending == 1) {
				$count = $pend_count;
				$pending = 0;
			} else
				$count = 0;

			$links = array();

			$result = mysql_query("select distinct link from
".$mysql_table_prefix."temp where level=$level && id='$sessid' order by link");
			echo mysql_error();
			$rows = mysql_num_rows($result);

			if ($rows == 0) {
				break;
			}

			$i = 0;

			while ($row = mysql_fetch_array($result)) {
				$links[] = $row['link'];
			}

			reset ($links);


			while ($count < count($links)) {
				$num++;
				$thislink = $links[$count];
				$urlparts = parse_url($thislink);
				reset ($omit);
				$forbidden = 0;
				foreach ($omit as $omiturl) {
					$omiturl = trim($omiturl);

					$omiturl_parts = parse_url($omiturl);
					if ($omiturl_parts['scheme'] == '') {
						$check_omit = $urlparts['host'] . $omiturl;
					} else {
						$check_omit = $omiturl;
					}

					if (strpos($thislink, $check_omit)) {
						printRobotsReport($num, $thislink,
$command_line);
```

```
                                        check_for_removal($thislink);
                                        $forbidden = 1;
                                        break;
                                }
                        }

                        if (!check_include($thislink, $url_inc, $url_not_inc )) {
                                printUrlStringReport($num, $thislink, $command_line);
                                check_for_removal($thislink);
                                $forbidden = 1;
                        }

                        if ($forbidden == 0) {
                                printRetrieving($num, $thislink, $command_line);
                                $query = "select md5sum, indexdate from
".$mysql_table_prefix."links where url='$thislink'";
                                $result = mysql_query($query);
                                echo mysql_error();
                                $rows = mysql_num_rows($result);
                                if ($rows == 0) {
                                        index_url($thislink, $level+1, $site_id, '',
$domain, '', $sessid, $can_leave_domain, $reindex);

                                        mysql_query("update
".$mysql_table_prefix."pending set level = $level, count=$count, num=$num where
site_id=$site_id");
                                        echo mysql_error();
                                }else if ($rows <> 0 && $reindex == 1) {
                                        $row = mysql_fetch_array($result);
                                        $md5sum = $row['md5sum'];
                                        $indexdate = $row['indexdate'];
                                        index_url($thislink, $level+1, $site_id, $md5sum,
$domain, $indexdate, $sessid, $can_leave_domain, $reindex);
                                        mysql_query("update
".$mysql_table_prefix."pending set level = $level, count=$count, num=$num where
site_id=$site_id");
                                        echo mysql_error();
                                }else {

        printStandardReport('inDatabase',$command_line);
                                }

                        }
                        $count++;
                }
                $level++;
        }
```

```
		mysql_query ("delete from ".$mysql_table_prefix."temp where id =
'$sessid'");
		echo mysql_error();
		mysql_query ("delete from ".$mysql_table_prefix."pending where site_id =
'$site_id'");
		echo mysql_error();
		printStandardReport('completed',$command_line);


	}

	function index_all() {
		global $mysql_table_prefix;
		$result=mysql_query("select url, spider_depth, required, disallowed,
can_leave_domain from ".$mysql_table_prefix."sites");
		echo mysql_error();
	while ($row=mysql_fetch_row($result)) {
		$url = $row[0];
			$depth = $row[1];
		$include = $row[2];
		$not_include = $row[3];
		$can_leave_domain = $row[4];
		if ($can_leave_domain=='') {
			$can_leave_domain=0;
		}
		if ($depth == -1) {
			$soption = 'full';
		} else {
			$soption = 'level';
		}
			index_site($url, 1, $depth, $soption, $include, $not_include,
$can_leave_domain);
		}
	}

	function get_temp_urls ($sessid) {
		global $mysql_table_prefix;
		$result = mysql_query("select link from ".$mysql_table_prefix."temp where
id='$sessid'");
		echo mysql_error();
		$tmp_urls = Array();
	while ($row=mysql_fetch_row($result)) {
			$tmp_urls[$row[0]] = 1;
		}
		return $tmp_urls;
```

```
	}

	function get_domains () {
		global $mysql_table_prefix;
		$result = mysql_query("select domain_id, domain from
".$mysql_table_prefix."domains");
		echo mysql_error();
		$domains = Array();
	while ($row=mysql_fetch_row($result)) {
			$domains[$row[1]] = $row[0];
		}
		return $domains;

	}

	function commandline_help() {
		print "Usage: php spider.php <options>\n\n";
		print "Options:\n";
		print " -all\t\t Reindex everything in the database\n";
		print " -u <url>\t Set url to index\n";
		print " -f\t\t Set indexing depth to full (unlimited depth)\n";
		print " -d <num>\t Set indexing depth to <num>\n";
		print " -l\t\t Allow spider to leave the initial domain\n";
		print " -r\t\t Set spider to reindex a site\n";
		print " -m <string>\t Set the string(s) that an url must include (use \\n as a
delimiter between multiple strings)\n";
		print " -n <string>\t Set the string(s) that an url must not include (use \\n as
a delimiter between multiple strings)\n";
	}

	printStandardReport('quit',$command_line);
	if ($email_log) {
		$indexed = ($all==1) ? 'ALL' : $url;
		$log_report = "";
		if ($log_handle) {
			$log_report = "Log saved into $log_file";
		}
		mail($admin_email, "Progress Report", "Crawler has finished indexing
$indexed at ".date("y-m-d H:i:s").". ".$log_report);
	}
	if ( $log_handle) {
		fclose($log_handle);
	}

?>
```

This crawler is similar to another PHP crawler called Sphider. The problem with Sphider's original code was it didn't have anything that prevented duplicated entries of a domain in the database, and it wouldn't allow multiple computers to crawl the domain at the same time. We used their basic crawler to guide our development of the 4x Web Crawler.

This web crawler is very simple. It performs the basic tasks needed to monitor how a basic search engine works. With this, we were able to see how a search engine starts in one domain, crawls the domain for all links, including links found when the user clicks on an image, follows those links and then proceeds to repeat infinitely. On top of that, it reads the content of the domain and returns basic keywords. It saves all this information into a database. There are many other aspects to the crawler as well, but to save space in this report, they are not being included. However, they are available to view upon request.

To prevent the search engine from displaying results that were inappropriate, a large list of keywords and phrases including foul language, sex references, drug references, and promotion of violence was created and if the crawler finds any of this information, it disregards it and doesn't save it to the database.

Verification

To verify the crawler was working properly, the Quemado Schools website was crawled. It was known that the website didn't have any inappropriate material so to see if the filter was working correctly, we copied the schools website into a local directory inside the crawler network and provided links on the homepage that lead to websites that were blocked at the school for inappropriate content. At this time, the crawler has some problems with certain phrases because of word choice, and spelling differences. If the website contains a phrase that is inappropriate but is mentioned in a short-speak type of way, the crawler cannot detect it. We are currently working on a way for users to report content. If the content is reported, that website will be put into a black list and must be reviewed by a network administrator. This stage is not working yet but we hope to have it working soon.

Verified Results

At the end of the project, we were able to conclude that a search engine is a very complicated system of files that work together to perform very complicated tasks automatically. The entire team now understands that a search engine looks at a domain, indexes the pages found on that domain for keywords and phrases that are related to that document, and moves on to index more domains found within the previous domain. In addition, popular search engines do not filter out inappropriate content. A likely reason is that they would not be a successful search engine if they regulated data. A search engine designed for schools has not been developed on a large scale yet because the internet is full of mixed content. Advertisements that change based on users make it complicated for search engines to monitor the entire site. Our crawler, even though it was not completely original, worked to our needs and was able to show how a search engine works. The code we wrote to prevent duplicated entries and entries with defined inappropriate keywords enhanced the crawler greatly. The final conclusion of this entire project is we have a lot of work to do. We did create a working model of a search engine that is dynamic, and it can change automatically based on seeds provided by users.

References

How Internet Search Engines Work: HowStuffWorks, Inc
<http://computer.howstuffworks.com/internet/basics/search-engine.htm>

PHP: w3 Schools
<http://www.w3schools.com/php/>

Original MySql API: PHP.net
<http://php.net/manual/en/book.mysql.php>

How Internet Filters Work: Infopeople.com
<http://infopeople.org/resources/filtering/how>

Sphyer: PHP Search Engine
<http://www.sphider.eu/>

Achievments

The primary achievement we all take pride in is the code that was able to filter out keywords and phrases. We had a lot of problems getting the search engine to read text as phrases and not individual words. After we worked through this problem, the code worked well and did exactly what we wanted it to do for the time restraints, hardware restraints, and other resource restraints we had.

Special Thanks

We would like to give a special thanks to Time Angelus. He was able to provide us with the hardware needed to run the search engine. Also, he worked hard to get us sponsored by Dream Spark.

In addition, we would also like to thank Dream Spark, a Microsoft program, for the software they gave us free of charge. They provided us with the operating systems we needed, and Visual Studios.