Crawler Architecture using Grid Computing

M.E.ElAraby[1], M.M.Sakre[1], M.Z.Rashad [2], O.Nomir [2]

1 Dept. of Computer Science,
High Institute for Computers and Information Systems,
Al Shorouk Academy, Egypt
{m_s_araby, m_sakre2001}@yahoo.com
2 Dept. of Computer Science, Faculty of Computer and Information Sciences,
Mansoura University, Egypt
{magdi_Z2011r, omnomi}@yahoo.com

ABSTRACT

Crawler is one of the main components in the search engines which use URLs to fetch web pages to build a repository of web pages starting with entering URL. Each web page is parsed to extract the URLs included in it and store the extracted URLs in the URLs Queue to fetch by the crawlers in sequential. The process of crawling takes long time to collect more web pages, and it has become necessary to utilize the unused computing resources and cost/time savings in organizations. This paper deals with the crawler of search engine using grid computing. This paper presents the grid computing that has been implemented by Alchemi. Alchemi is an open source project developed at the University of Melbourne, provides middleware for creating an enterprise grid computing environment. The crawling processes are passed to Alchemi manager which distribute the processes over a number of computers as executors. The search engine crawler with the grid computing is implemented, tested and the results are analyzed. There is an increase in performance and less time over the single computer.

Keywords:

Crawler, URL, Grid Computing, Alchemi, Manager, Executor, performance, and web pages,

1. Introduction

The internet is one of the main sources of information for a large number of consumers. Thus, search engines as the mediators between online information and consumers are an important topic of research. A search engine has at least three main components [1] as shown in figure (1). The roll of the search engine is to gathering the web pages and indexing them to retrieve easily. One of the main components in search engines, which fetch the web pages, is the job of crawler.

DOI : 10.5121/ijcsit.2012.4310

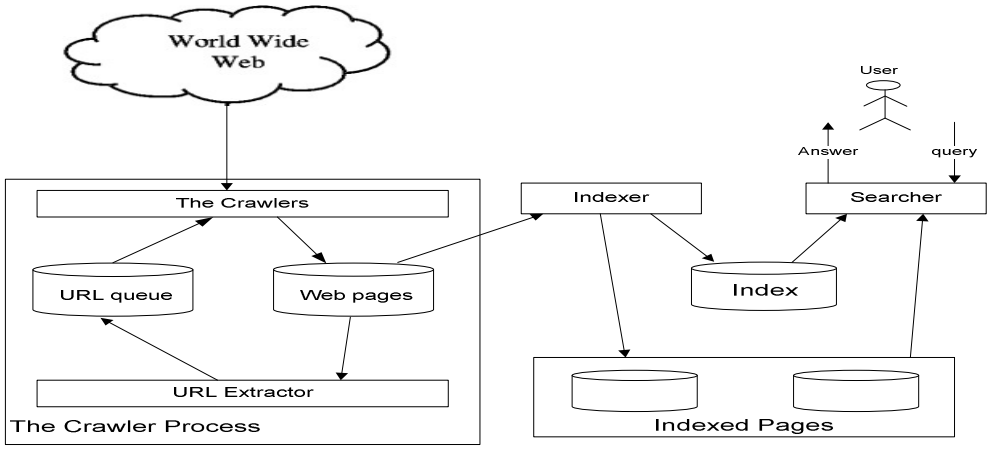


Figure (1) Search Engine Generic Architecture

Web pages are written in a tagged markup language called the hypertext markup language (HTML). A hyperlink is expressed as an anchor tag with a href attribute, names another page using a uniform resource locator (URL). The only way to collect URLs is to scan collected pages for hyperlinks to other pages that have not been collected yet. This is the basic principle of crawlers. They start from a given set of URLs, progressively fetch and scan them for new URLs and then fetch these pages in turn, in an endless cycle [2].

A single page fetch may involve several seconds of network latency, it is essential to fetch many pages at the same time to utilize the network bandwidth available. Extract URLs and eliminate duplicates to reduce redundant fetches and to avoid spider traps.

The crawler downloads a series of pages whose sizes in bytes are P_i, and has B bytes per second of available bandwidth for doing it, then it should download all the pages simultaneously at a speed proportional to the size of each page:

$$T^* = \frac{\sum P_i}{B}$$

T^* is the optimal time to use all the available bandwidth. This optimal scenario is depicted in Figure 2.

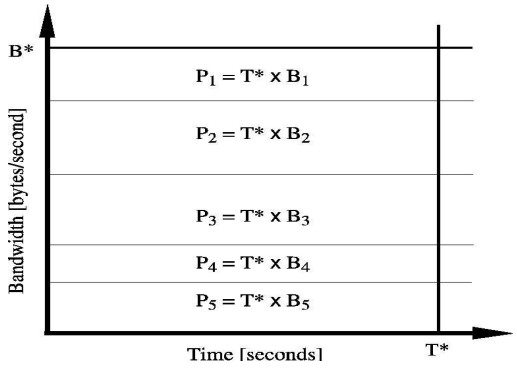


Figure (3) Optimal scenario for downloads.

However, there are many restrictions that forbid this optimal scenario. The main restriction is that it must avoid overloading Web sites: a Web crawler can impose a substantial amount of work on a Web server, specially if it opens many simultaneous connections for downloading [14]. The crawler does not download more than one page from the same Web site at a time, and it waits between 30 and 60 seconds between accesses. This, together with the fact that Web sites have usually a bandwidth B_i^{MAX} that is lower than the crawler bandwidth B, originate download time lines similar to the one shown in figure 3. The optimal time T* is not achieved, because some bandwidth is wasted due to limitations in the speed of Web sites and to the fact that we must wait between accesses to a Web. There is another serious practical restriction: the HTTP request has latency, and the latency time can be over 25% of the total time of the request [15].

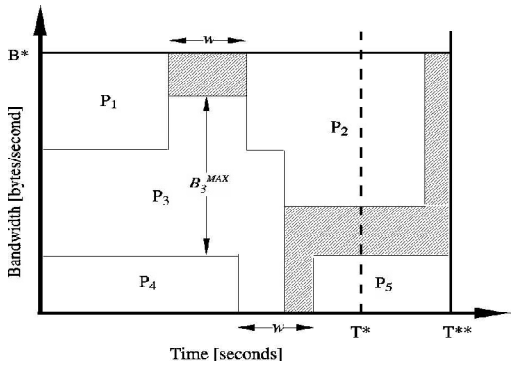


Figure (4) more realistic scenario; the hatched portion is wasted bandwidth.

There is a large change in the way of perceiving, using computing in the last ten years. It was normal to perform computing needs to be processed by localized computing platforms and infrastructures. This way has been changed. The change has been caused by the take-up of commodity computer and network components. A consequence of these changes has been the capability for effective and efficient utilization of widely distributed resources to fulfill a range of application needs [3, 4, 5].

When computers are interconnected and communicating, we have distributed system, and the issues in designing, building and deploying distributed computer systems have been discussed in many years. The Grid computing term was known in the mid1990s that refers to a proposed distributed computing infrastructure for advanced science and engineering [7, 8]. In general, the grid computing term is a special type of parallel computing that relies on complete computers (with onboard CPU, storage, power supply, network interface, etc.) connected to a network (private, public or the Internet) by a conventional network interface, such as Ethernet. A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [9].

2. Related work

There are a number of research groups that have been working in the field of distributed computing. These groups have created middleware, libraries and tools that allow the cooperative use of geographically distributed resources unified to act as a single powerful platform for the execution of parallel and distributed applications. This approach of computing has been known by several names, such as metacomputing, scalable computing, global computing, Internet computing and lately as grid computing [6, 7, 8].

There are many research studies in web crawling, such as URL ordering for retrieving high-quality pages earlier [16], partitioning the web for efficient multi-processor crawling [17], distributed crawling [18], and focused crawling [19]. There are some crawling architectures that designed based grid computing that aims to increase the performance in specific issues in web crawling, such as a grid focused community crawling architecture for medical information retrieval services [20], Multi Agent System-based crawlers for Virtual Organizations [21], a dynamic URL assignment method for parallel web crawler [22], A Middleware for Deep Web Crawling Using the Grid [23], and Architecture of a grid-enabled Web search engine [24].

3. Alchemi Tool

Alchemi system is one of the systems that help in implementing the grid based applications. It is supported by Microsoft windows-based grid computing infrastructure that plays critical role in the industry-wide adoption of grids due to the large-scale deployment of windows within enterprises. Alchemi runs on the Windows operating system in the .NET Framework [10]. Alchemi provides an API for C# and C++, and operates in the Windows .NET framework, so we use C# APIs to develop.

Alchemi system is open source software toolkits developed at the University of Melbourne, which provides middleware for creating an enterprise grid computing environment. Alchemi consists of two main components. These are the manager and the executer components. As it is explained in figure (4), the executor component can be run on many computers while only one computer runs the manager component. The manager receives the threads from the client application and distributes these threads over the connected executors. The manager stores the execution time and the executor of each thread [11].

While the notion of grid computing is simple enough, the practical realization of grids poses a number of challenges. Key issues that need to be dealt with are security, heterogeneity, reliability, application composition, scheduling, and resource management. The Microsoft .NET Framework provides a powerful toolset that can be leveraged for all of these, in particular support for remote execution, multithreading, security, asynchronous programming, disconnected data access, managed execution, and cross-language development, making it an ideal platform for middleware grid computing [12].

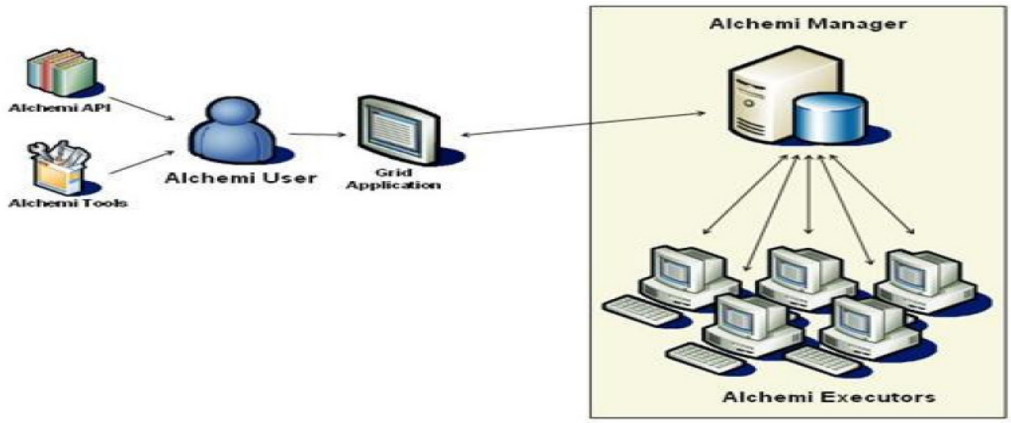


Figure (5) Alchemi's main components

4. The Crawler Architecture

The crawler consists of multiple processes which send requests and receives responses of the requests as pages. It parses these pages to extract the URLs contained in the pages and stores them in the URLs Queue; each URL in the queue is crawled and so on. The aim of these processes is to collect more pages and extracts more URLs from these pages to create a large repository of URLs and pages from web. These processes execute large number of process and consume large time. So, a modification is suggested to distribute the processes of the crawler into threads and use multiple computers to execute these threads in balancing, to distribute the efforts and minimize the time of the crawling execution. The block diagram in figure (5) shows the proposed architecture of the crawler system, using grid computing.

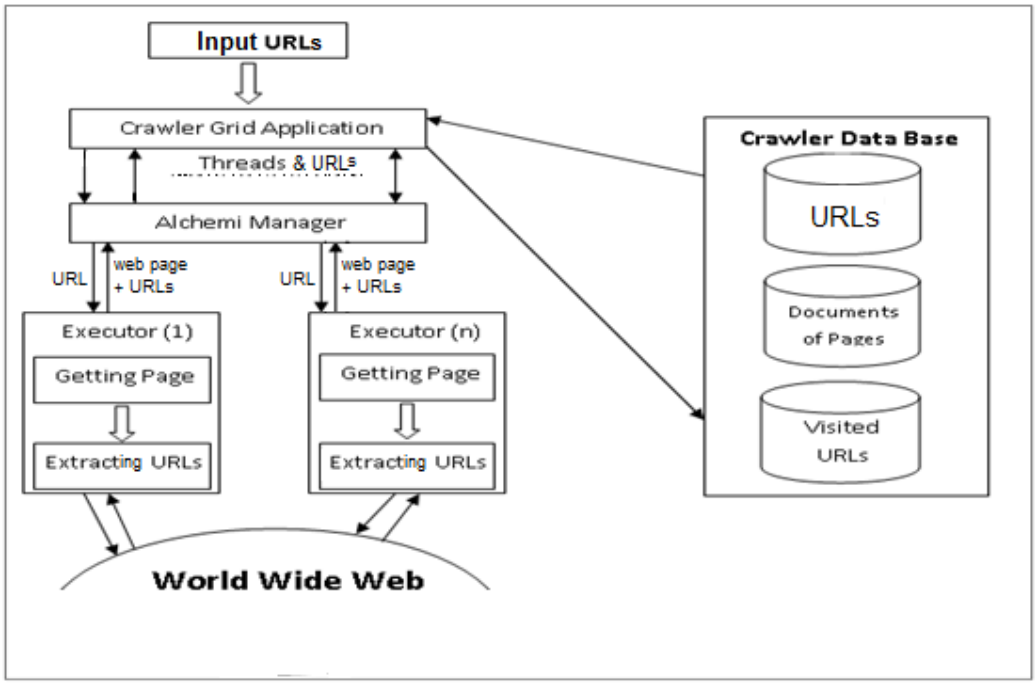


Figure (6) flow of the crawler application

This architecture comes based on concept lies in parallel / distributed processing where the goal is twofold; reducing the overall elapsed processing time of collecting the same amount of pages, and utilizing the unused computing resources (processing power). There is an important classification of grid that classifies it into compute grid and utility grid. Compute grid generally has an application that requires intense computing actions and that could be divided into subtasks that could run in parallel and combined later to yield the desired result. These sub-tasks are executed in different machines in order to ensure time effectiveness and resource utilization. Utility grid on the other hand is a collection of hardware resources that is virtualized so that multiple users can submit their individual applications to this pool of resources and have the benefit of load balancing, maximum use of resources, and optimal turnaround time for each individual application [13].

In the block diagram first step input the URL, the crawler grid application generates a number of threads. The threads are passed to Alchemi manager which distributes these threads over the available executors connected to it, and the results are returned in the reverse direction.

5. Implementation

As Alchemi system provides a Software Development Kit (SDK) that includes a Dynamic Link Library (DLL) that supports multithreaded applications, the proposed architecture of the crawler grid application, as in figure (5) is implemented.

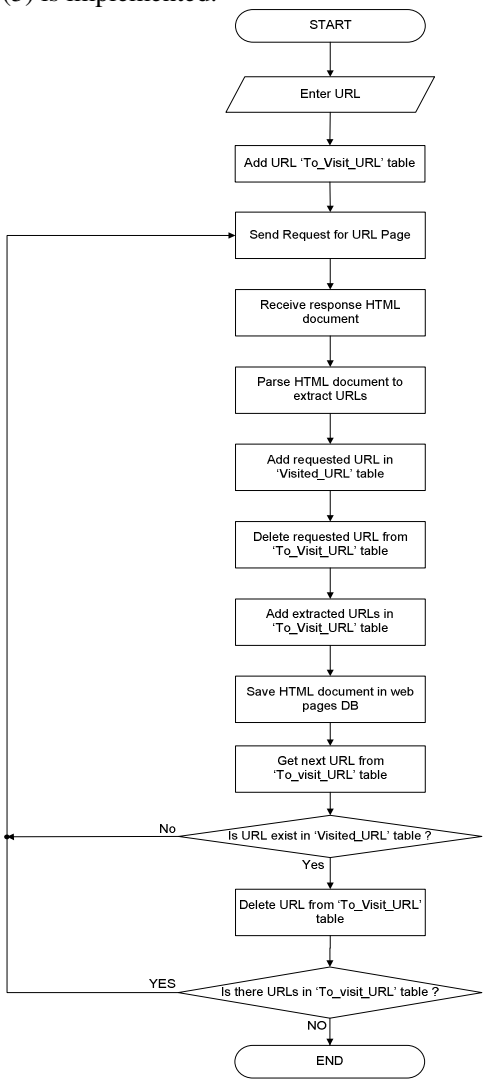


Figure (7) flow of the crawler application

The flowchart of the normal crawler application is shown in figure (6) in which the crawling processes are sequentially executed on single executor. Where the crawling steps as the following:

1- Start crawling.
2- Entering URL.
3- Add the URL in a queue to_visit_URL.

4- Send request for URL page.
5- Receive response html document.
6- Parse HTML document to extract URLs.
7- Add requested URL in visited_URL queue.
8- Delete requested URL from to_visit_URL queue.
9- Add extracted URLs in to_visit_URL queue.
10- Save HTML document in web pages DB.
11- Get next URL from to_visit_URL queue.
12- IF URL exist in visited_URL queue:
    Then delete URL from to_visit_URL queue.
    Else go to step 3.
13- If there is URLs in to_visit_URL queue:
    Then go to step 3.
    Else end Crawling.

When, grid computing architecture is used, the flow chart of the crawler grid application is shown in figure (7), in which the processes are executed in multithreading methodology. Where the crawling steps are running in three layers: the application machine, Alchemi manager, Alchemi executor.the steps are distributed over these layers as the following:
The application machine:

1- Start.
2- Enter URL.
3- Add URL in the to_visited_URL queue.
4- Create thread with URL.
5- Pass the thread to alchemi manager.
6- Receive result of thread execution manager.
7- Add requested URL in visited_URL queue.
8- Delete requested URL from to_visit_URL queue.
9- Add extracted URLs in to_visit_URL queue.
10- Save HTML document in web pages DB.
11- Get next URL from to_visit_URL queue.
12- IF URL exist in visited_URL queue:
    Then delete URL from to_visit_URL queue.
    Else go to step 3.
13- If there is URLs in to_visit_URL queue:
    Then go to step 3.
    Else end Crawling.

Alchemi Manager:

1- Get thread from application.
2- Assign thread to an executor.
3- Receive results of finished thread.
4- Send results to the application.
5-

Alchemi Executors:

1- Get thread from manager.
2- Send request RUL page.
3- Receive response HTML document.
4- Parse HTML document to extract URLs.
5- Send results back to manager.

The programming model of the crawler grid application is shown in figure (8); the system model is distributed using a programming model as multi-tier implementation approach. The crawler is composed of four tiers. The 4-tier architecture: The user interface, Control, Data Storage, Executors.

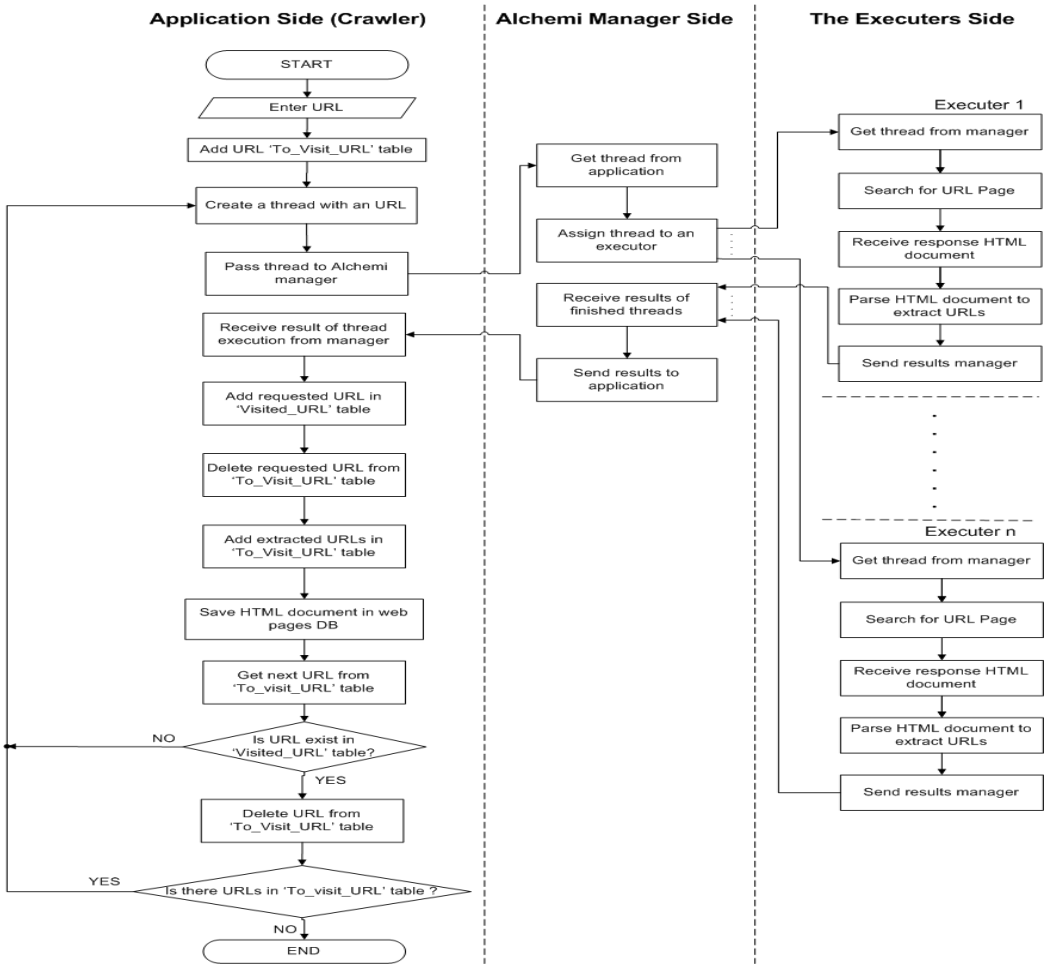


Figure (8) the crawler grid application

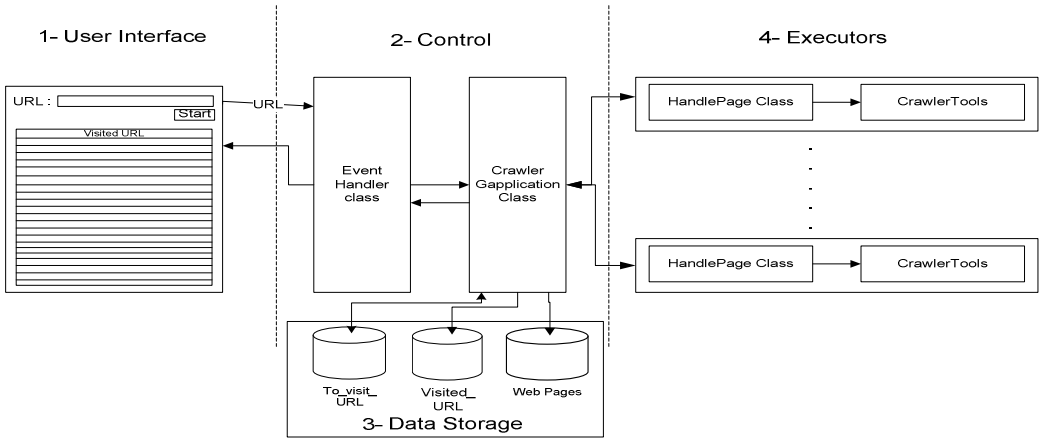


Figure (9) programming module

The user interface layer is designed to input the list of initial seed URLs.

The control layer is the main core of the system architecture; it has been implemented using C# language. This layer contains the Crawler class is called by the event handler. This class contains a method that create Grid Connection 'GConnection' object that set connection to Alchemi manager by passing the manager IP, port number, username, and password to it, then create Grid Application 'GApplication' object and instantiate an object of a Grid Thread 'GThread' class then add the Grid Thread object to the Grid Application.

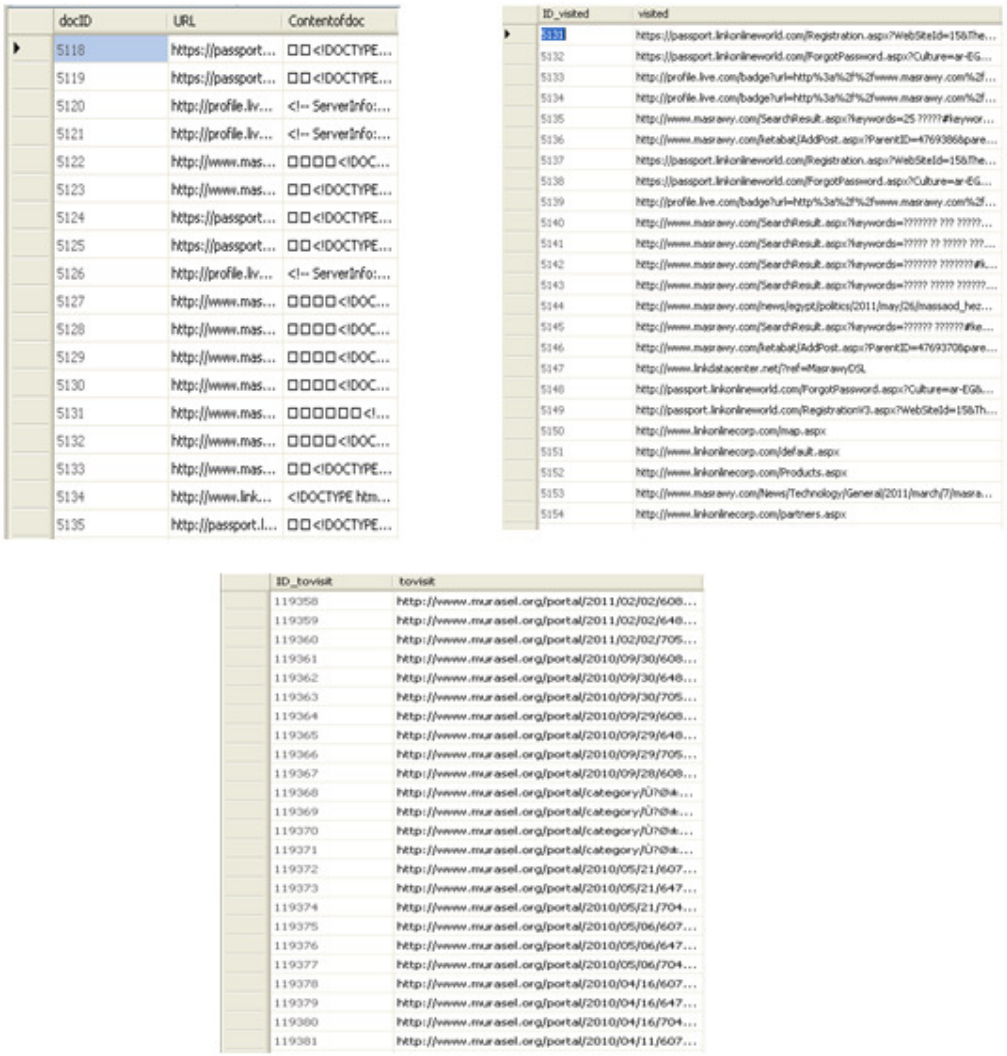
```
// get settings from user
GConnection gc = GConnection.FromConsole("192.168.10.8", "9000",
"user", "user");
// create a new grid application
GApplication App = new GApplication(gc);
// add the module containing PiCalcGridThread to the application
manifest
                App.Manifest.Add(new
ModuleDependency(typeof(crawlerISA.Handle_page).Module));

for (int i = 0; i < to_visit_urls.Count; i++)
{
Handle_page  hp = new     Handle_page(to_visit_urls[0], ref
visited_urls, ref to_visit_urls);
}
```

The executor layer contains the HandlePage class which extends Gthread class and overrides start method. In start method instantiate an object of CrawlerTools class which contain two main methods the first called crawl_page that take the URL and return the content page, the second method called parse_page that take the page content and return list of URLs contained in that page.

```
public class CrawlerTools
{
/* instance passing list by reference */
public string crawl_page(string URl, ref List<string> visited_urls,
ref List<string> to_visit_urls)
.
.
/* instance passing list by reference*/
public List<string> parse_page(string pageContent, ref List<string>
visited_urls, ref List<string> to_visit_urls)
.
.
```

The execution details of all threads are stored in Alchemi database which contains table called Executors that contains the details of all executors connected to the manager, table called Applications that contains the details of all applications that runs on the manager, and table called threads that contains the details of all threads the executed through the manager and store the executor that execute the thread.

The data storage layer contains the links that will request, the links that have been requested and the web pages that have been collected and stored in the database, it is designed a simple database in SQL Server Database Engine to store these data as shown in Figure (9), this database consists of three tables, the first table is document table that contain the HTML document of each fetched page, the URL of this page and identifier number unique (docID), the second table is to_visit_URL table that contain the URLs extracted of the HTML document, the third table is visited_URL table that contain the URLs that have been requested previously. The database is designed specially to serve the crawling process only in the search engine without any consideration to the other components in the search engine as indexing and searcher program.



docID	URL	Contentofdoc
5118	https://passport...	□□<!DOCTYPE...
5119	https://passport...	□□<!DOCTYPE...
5120	http://profile.liv...	<!-- Server1nfo:...
5121	http://profile.liv...	<!-- Server1nfo:...
5122	http://www.mas...	□□□□<!DOC...
5123	http://www.mas...	□□<!DOCTYPE...
5124	https://passport...	□□<!DOCTYPE...
5125	https://passport...	□□<!DOCTYPE...
5126	http://profile.liv...	<!-- Server1nfo:...
5127	http://www.mas...	□□□□<!DOC...
5128	http://www.mas...	□□□□<!DOC...
5129	http://www.mas...	□□□□<!DOC...
5130	http://www.mas...	□□□□<!DOC...
5131	http://www.mas...	□□□□□□<!...
5132	http://www.mas...	□□□□<!DOC...
5133	http://www.mas...	□□<!DOCTYPE...
5134	http://www.link...	<!DOCTYPE htm...
5135	http://passport.l...	□□<!DOCTYPE...

ID_visited	visited
5131	https://passport.linkonlineworld.com/Registration.aspx?WebSiteId=158The...
5132	https://passport.linkonlineworld.com/ForgotPassword.aspx?Culture=ar-EG...
5133	http://profile.live.com/badge?url=http%3a%2f%2fwww.masrawy.com%2f...
5134	http://profile.live.com/badge?url=http%3a%2f%2fwww.masrawy.com%2f...
5135	http://www.masrawy.com/SearchResult.aspx?keywords=25 ?????#keywor...
5136	http://www.masrawy.com/ketabat/AddPost.aspx?ParentID=47693868spare...
5137	https://passport.linkonlineworld.com/Registration.aspx?WebSiteId=158The...
5138	https://passport.linkonlineworld.com/ForgotPassword.aspx?Culture=ar-EG...
5139	http://profile.live.com/badge?url=http%3a%2f%2fwww.masrawy.com%2f...
5140	http://www.masrawy.com/SearchResult.aspx?keywords=??????? ??? ?????...
5141	http://www.masrawy.com/SearchResult.aspx?keywords=????? ?? ????? ???...
5142	http://www.masrawy.com/SearchResult.aspx?keywords=??????? ???????#k...
5143	http://www.masrawy.com/SearchResult.aspx?keywords=????? ????? ??????...
5144	http://www.masrawy.com/news/egypt/politics/2011/may/26/massaod_hez...
5145	http://www.masrawy.com/SearchResult.aspx?keywords=?????? ??????#ke...
5146	http://www.masrawy.com/ketabat/AddPost.aspx?ParentID=47693708spare...
5147	http://www.linkdatacenter.net/?ref=MasrawyDSL
5148	http://passport.linkonlineworld.com/ForgotPassword.aspx?Culture=ar-EG&...
5149	http://passport.linkonlineworld.com/Registration#3.aspx?WebSiteId=158Th...
5150	http://www.linkonlinecorp.com/map.aspx
5151	http://www.linkonlinecorp.com/default.aspx
5152	http://www.linkonlinecorp.com/Products.aspx
5153	http://www.masrawy.com/News/Technology/General/2011/march/7/masra...
5154	http://www.linkonlinecorp.com/partners.aspx

ID_tovisit	tovisit
119358	http://www.murasel.org/portal/2011/02/02/608...
119359	http://www.murasel.org/portal/2011/02/02/648...
119360	http://www.murasel.org/portal/2011/02/02/705...
119361	http://www.murasel.org/portal/2010/09/30/608...
119362	http://www.murasel.org/portal/2010/09/30/648...
119363	http://www.murasel.org/portal/2010/09/30/705...
119364	http://www.murasel.org/portal/2010/09/29/608...
119365	http://www.murasel.org/portal/2010/09/29/648...
119366	http://www.murasel.org/portal/2010/09/29/705...
119367	http://www.murasel.org/portal/2010/09/28/608...
119368	http://www.murasel.org/portal/category/Ù?Ø%...
119369	http://www.murasel.org/portal/category/Ù?Ø%...
119370	http://www.murasel.org/portal/category/Ù?Ø%...
119371	http://www.murasel.org/portal/category/Ù?Ø%...
119372	http://www.murasel.org/portal/2010/05/21/607...
119373	http://www.murasel.org/portal/2010/05/21/647...
119374	http://www.murasel.org/portal/2010/05/21/704...
119375	http://www.murasel.org/portal/2010/05/06/607...
119376	http://www.murasel.org/portal/2010/05/06/647...
119377	http://www.murasel.org/portal/2010/05/06/704...
119378	http://www.murasel.org/portal/2010/04/16/607...
119379	http://www.murasel.org/portal/2010/04/16/647...
119380	http://www.murasel.org/portal/2010/04/16/704...
119381	http://www.murasel.org/portal/2010/04/11/607...

Figure (10)Database has been designed to store the documents, the visited URLs, and URLs to visit.

6. Results and Evaluation

The used testbed consists of five personal computers with 1GB RAM and Intel CPU 2.3 GHz all the five PCs connected with an internet Line, Alchemi executer program is installed on each computer as in Figure (10), and Alchemi manager program is installed on a separate computer as in Figure (11), where SQL server 2008 is installed. Alchemi executor program is installed on the other computers with Microsoft .NET framework 1.1.

Run the manager node and one or more executor nodes and connect them to the manager node that is configured when constructing a desktop grid in Alchemi as show in Figure (12), the figure shows the executer table in the Alchemi database which store the details of each executer. The threads that are submitted to the manager stored in the threads table in the alchemi database as in Figure (13).

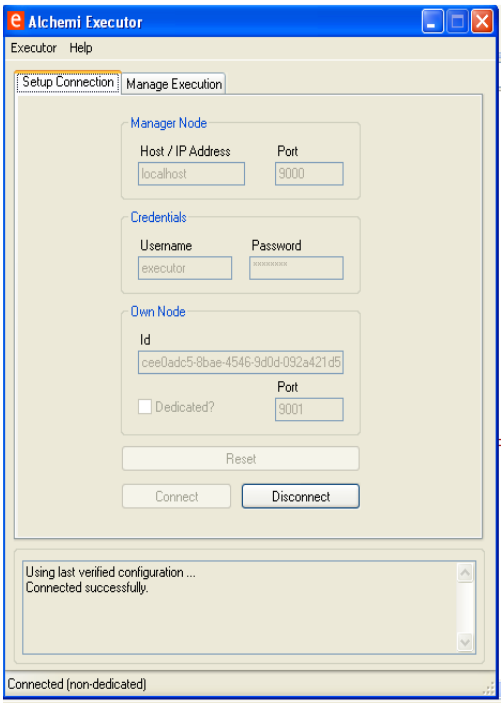


Figure (11) Alchemi Executor Form

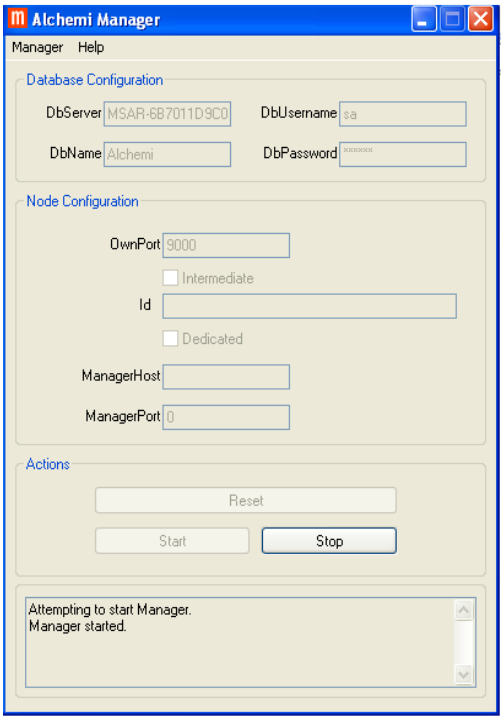


Figure (12) Alchemi Manager Form

	executor_id	is_dedicated	is_connected	ping_time	host	port	usr_name	cpu_max	cpu_usage	cpu_avail	cpu_totalusage
1	B2F88439-61F3-491F-A4B9-275639BFCB1C	1	1	2012-02-14 14:45:02.023	pc4	9001	executor	2294	3	95	151.06
2	7CC22710-CC3D-46CA-8466-8CDD1D3410BC	1	1	2012-02-14 14:44:36.413	pc3	9001	executor	2294	0	17	159.86
3	841D0C66-A802-4763-AC85-B68F773FDB98	1	1	2012-02-14 15:26:24.460	pc2	9001	executor	2294	0	27	111.3
4	9084E6BA-8ABF-490F-A9BB-E50636C383BB	1	1	2012-02-14 14:44:57.163	pc6	9001	executor	2295	3	51	198.780000000001

Figure (13) Achemi Executors connected to manager

	internal_thread_id	application_id	executor_id	thread_id	state	time_started	time_finished	priority
1	367761	9BA0FE17-41BB-403D-9F...	B2F88439-61F3-49...	0	4	2012-02-14 14:23:56.697	2012-02-14 14:24:01.977	5
2	367762	9BA0FE17-41BB-403D-9F...	7CC22710-CC3D-4...	1	4	2012-02-14 14:23:57.057	2012-02-14 14:24:12.133	5
3	367763	9BA0FE17-41BB-403D-9F...	841D0C66-A802-4...	2	4	2012-02-14 14:23:56.807	2012-02-14 14:24:22.273	5
4	367764	9BA0FE17-41BB-403D-9F...	9084E6BA-8ABF-4...	3	4	2012-02-14 14:23:56.617	2012-02-14 14:24:11.760	5
5	367765	9BA0FE17-41BB-403D-9F...	B2F88439-61F3-49...	4	4	2012-02-14 14:24:02.010	2012-02-14 14:24:09.977	5
6	367766	9BA0FE17-41BB-403D-9F...	B2F88439-61F3-49...	5	4	2012-02-14 14:24:09.977	2012-02-14 14:24:15.790	5
7	367767	9BA0FE17-41BB-403D-9F...	9084E6BA-8ABF-4...	6	4	2012-02-14 14:24:11.773	2012-02-14 14:24:15.790	5
8	367768	9BA0FE17-41BB-403D-9F...	7CC22710-CC3D-4...	7	4	2012-02-14 14:24:12.180	2012-02-14 14:24:18.570	5
9	367769	9BA0FE17-41BB-403D-9F...	B2F88439-61F3-49...	8	4	2012-02-14 14:24:15.883	2012-02-14 14:24:21.053	5
10	367770	9BA0FE17-41BB-403D-9F...	9084E6BA-8ABF-4...	9	4	2012-02-14 14:24:15.993	2012-02-14 14:24:23.070	5

Figure (14) The Threads table in the alchemi database

The crawler grid application is on the computer which contains Alchemi manager, several experiments are on the executors and collecting 50, 100, 150, 200, 250, 300 pages and record the results of each experiment as in table 1.
Now, if only one CPU is available then the total time is

$$T = X * (t_c + t_e)$$

X Be number of pages,

t_c Time of receiving one page,

t_e Time of extracting the URLs in the page,

But if n nodes are available, then total time is

$$T = \frac{X}{N} * (t_c + t_e)$$

N Be the number of executor nodes.

Thus, by increasing the number of Executor nodes, the time decreases in a linear fashion, proportionally to the number of executor nodes in the Grid. This is evident in the results shown in Table 1 and the graph in Figure (14).

Thus, there will be a great improvement in performance when increasing the number of executor nodes. It is to be noted that there is no consideration to network overload and the difference in page sizes.

Table 1 Time of crawling pages that have been collected

no. of Pages	50	100	150	200	250	300
1 Executor	4.8	8	13.2	21.2	24.21	36.12
2 Executor	2.37	3.49	6.54	10.44	12	18.01
3 Executor	1.77	2.61	4.90	7.82	8.99	13.5
4 Executor	1.18	1.72	3.22	5.15	5.91	8.89
5 Executor	1.06	1.61	2.79	4.54	5.18	7.24

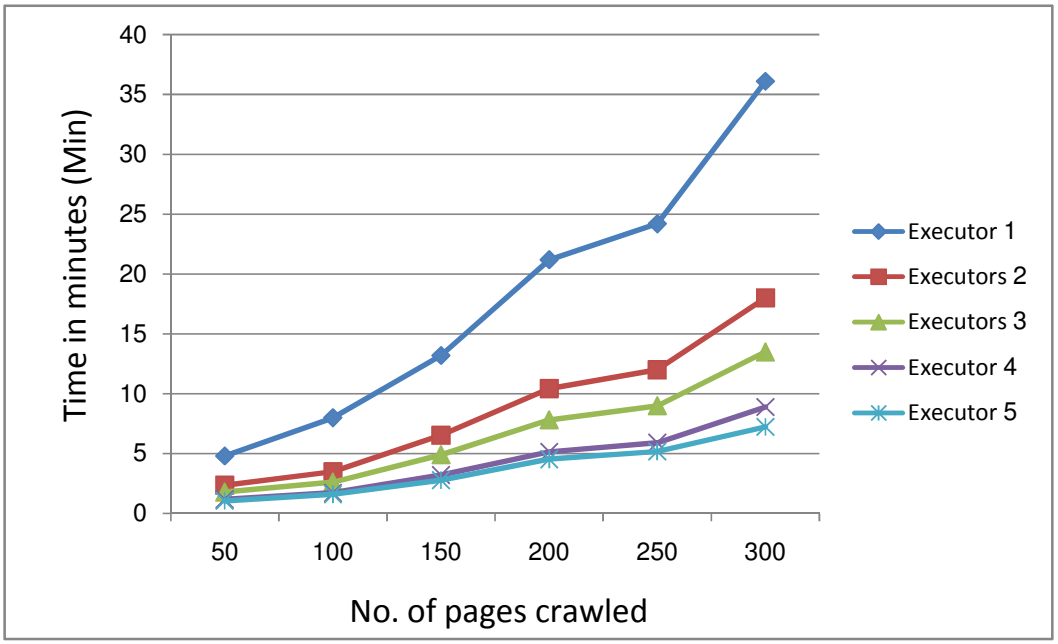


Figure (15) Execution time using varying number of executors enabled.

7. Conclusion and Future Work

The implementation of the crawler in search engine requires powerful computers to achieve high performance. This paper shows that the crawler performance can be enhanced by increasing the number of executor nodes. So, the required time to collect a greater number of pages is minimized. This evolution in performance is achieved by aggregating the power of distributed resources available in a network using grid scheduling system. This cooperation of the execution of the crawler application is achieved by using an open source application called Alchemi with its two main components, the manager and executor, to distribute and execute jobs.

The main aim of this paper is to improve the crawler efficiency, in collecting web pages, by better utilization of the available resources. It is planned, as a future work to implement the crawler component which updates the collected web pages regularly. Grid computing may be used to enhance the performance of other search engine components such as Page Ranking.

References:

[1] Sergey Brin and Lawrence Page "The Anatomy of a Large-Scale Hypertextual Web Search Engine", Computer Science Department, Stanford University, Stanford, CA 94305 : http://infolab.stanford.edu/~backrub/google.html .

[2] Soumen Chakrabarti, Book: "Mining the web, Discovering knowledge from hypertext data"; August 15, 2002.

[3] Ahmar Abbas, Book: "GRID COMPUTING: A Practical Guide to Technology and Application", ISBN: 1-58450-276-2, Charles River Media Inc, 2004.

[4] http://www.gridcomputing.com/.

[5] http://www.gridforum.org/.

[6] Ian Foster, Carl Kesselman, Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations, International Journal of High Performance Computing Applications Fall 2001 15: 200-222.

[7] Ian Foster, Carl Kesselman, Jeffrey M. Nick, Steven Tuecke, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Global Grid Forum, June 22, 2002.

[8] Fran Berman, Geoffrey Fox, Tony Hey. Book: Grid Computing: Making the Global Infrastructure a Reality, published March 2003.

[9] Foster, I. and Kesselman, C. (eds.). The Grid2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1999.

[10] Akshay Luther et al, “Peer-to-peer grid computing and a .NET-based Alchemi framework”, High performance computing: paradigm and infrastructure, Laurence Yang and Minyi Guo (eds), Chap 21, 403-429, Wiley Press, New Jersey, USA, June 2005.

[11] Akshay Luther, Rajkumar Buyya, Rajiv Ranjan & Srikumar Venugopal, Alchemi: A .NET-based Grid Computing Framework and its Integration into Global Grids, Technical Report, GRIDS-TR-2003-8, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia.

[12].Krishna Nadiminti, Yi-Feng Chiu, Nick Teoh, Akshay Luther, Srikumar Venugopal, and Rajkumar Buyya, ExcelGrid: A .NET Plug-in for Outsourcing Excel Spreadsheet Workload to Enterprise and Global Grids, Proceedings of the 12th International Conference on Advanced Computing and Communication (ADCOM 2004, December 15-18, 2004), Ahmedabad, India.

[13] SAS Global Forum 2011, Designing a Grid Computing Architecture: A Case Study of Green Computing Implementation Using SAS®, Krishnadas N, Indian Institute of Management, Kozhikode, Kerala, India, Paper 366-2011.

[14] Martijn Koster. Robots in the web: threat or treat ? ConneXions, 9(4), April 1995.

[15] B. Liu and E. A. Fox. Web traf_c latency: Characteristics and implications. J.UCS: Journal of Universal Computer Science, 4(9):763.778, 1998.

[16] Baeza-Yates, Castillo, Marin, & Rodriguez, 2005; Cho, Garcia-Molina, & Page, 1998; Najork &Wiener, 2001.

[17] Cambazoglu et al., 2004; Teng, Lu, Eichstaedt, Ford, & Lehman, 1999.

[18] Boldi, Codenotti, Santini, & Vigna, 2002; Zeinalipour-Yazti & Dikaiakos, 2002.

[19] Altingovde & Ulusoy, 2004; Chakrabarti, van den Berg, & Dom, 1999; Diligenti, Coetzee, Lawrence, Giles, & Gori, 2000.
[20] K. Cerbioni, E. Palanca, A. Starita, F. Costa, P. Frasconi, Conf. on Computational Intelligence in Medicine and Healthcare, CIMED'2005.
[21] Mohammed Ben-Mubarak; Azrul Hasni, Ian Chai, "Multi Agent System-based crawlers for Virtual Organizations ", IEEE International Conference of Distributed Frameworks for Multimedia Applications, May 2006.
[22] Guerriero, A., Ragni, F., Martines, C., "A dynamic URL assignment method for parallel web crawler", IEEE International Conference of CIMSA, on pages 119 – 123, Sept. 2010.
[23] Jihwan Song, Dong-Hoon Choi, Yoon-Joon Lee, " OGSA-DWC: A Middleware for Deep Web Crawling Using the Grid ", IEEE Fourth International Conference of eScience, on pages 370 - 371, Dec. 2008.
[24] B. Barla Cambazoglu, Evren Karaca, Tayfun Kucukyilmaz, Ata Turk, Cevdet Aykanat, " Architecture of a grid-enabled Web search engine", Available online 11 December 2006, Information Processing and Management number 43 on pages 609–623, 2007.