Dynamic and Distributed Indexing Architecture in Search Engine using Grid Computing

M. E. ElAraby
Dept. of Computer Science, High Institute for Computers and Information Systems,
Al Shorouk Academy, Egypt

M. M. Sakre
Dept. of Computer Science, High Institute for Computers and Information Systems,
Al Shorouk Academy, Egypt

M. Z. Rashad
Dept. of Computer Science, Faculty of Computer and Information Sciences,
Mansoura University, Egypt

O. Nomir
Dept. of Computer Science, Faculty of Computer and Information Sciences,
Mansoura University, Egypt

ABSTRACT
Search engines require computers with high computation resources for processing to crawl web pages and huge data storage to store billions of pages collected from the World Wide Web after parsing and indexing these pages. The indexer is one of the main components of the search engine that come intermediate between the crawler and the searcher. Indexing is the process of organizing the collected data to facility information retrieval and minimizes the time of query. Indexing requires huge processing and storage resources, and the indexing has a high effect on the performance of the search engine, this effect differs based on the structure and the process index construction. Distribution of the indexing process over a cluster of computers in grid computing will improve the performance through distributing the parsing load over a number of computers in a grid environment, and distributing the indexed data over distributed memory according to terms over a number of computers remotely. Due to the search engine data collections with frequent changes, the indexer require dynamic indexing. So the merge of the distributed and dynamic indexing in architecture over grid computing will give a better performance utilizing the available resources without need to computers with high cost such as supercomputers.

General Terms
Grid Computing, Algorithms, Inverted Index, and World Wide Web.

Keywords
Indexer, World Wide Web, Search engine, Grid Computing, Web pages, Secondary index, Main index, Alchemi, Manager, and Executor.

1. INTRODUCTION
One of the main sources of information is the internet which is useful for a large number of consumers. The easiest way to deal with the internet is the search engine. Search engines work as the mediators between consumers and online information to search and get information from the World Wide Web which contains many billions of web pages. So, components of search engine are an important topic of research. A search engine has at least three main components: the crawler, indexer, and searcher [1] as shown in figure 1. The search engine roll is to gathering the web pages and indexing them to retrieve easily and faster by user queries. One of the main components in search engines is the indexing which consists of steps followed to generate the indexed pages collection.

Indexing is the act of classifying and providing an index in order to make items easier to retrieve. Indexing the data set enables access to large data set, and reduces the access time while query processing by the way to avoid linearly scanning the texts for each query is to index the documents in advance. The crawling in the search engine collects very large number of web pages in database, then when search query come, it will take much time to retrieve the required page, so indexing process in search engine parses, and stores data to facilitate fast and accurate information retrieval. There are many different indexing techniques and each one has different performance and speed. There are two general types of indexing full-text indexing, and partial-text indexing. Full-text indexing parses and store all words in the document so it requires more storage and increases index size, but partial-text services restrict the depth indexed to reduce index size. Popular search engines focus on the full-text indexing of online, natural language documents. [2]

Indexing in search engine is one of the main components of the search engine, it is an important factor of search engine, and it is intermediate stage between crawling process and the searching, so the indexing affects the general performance, speed, and accuracy of the search engine. The performance of search engine and underlying indexing techniques is one of the factors that a critical for usability of text retrieval systems [3].

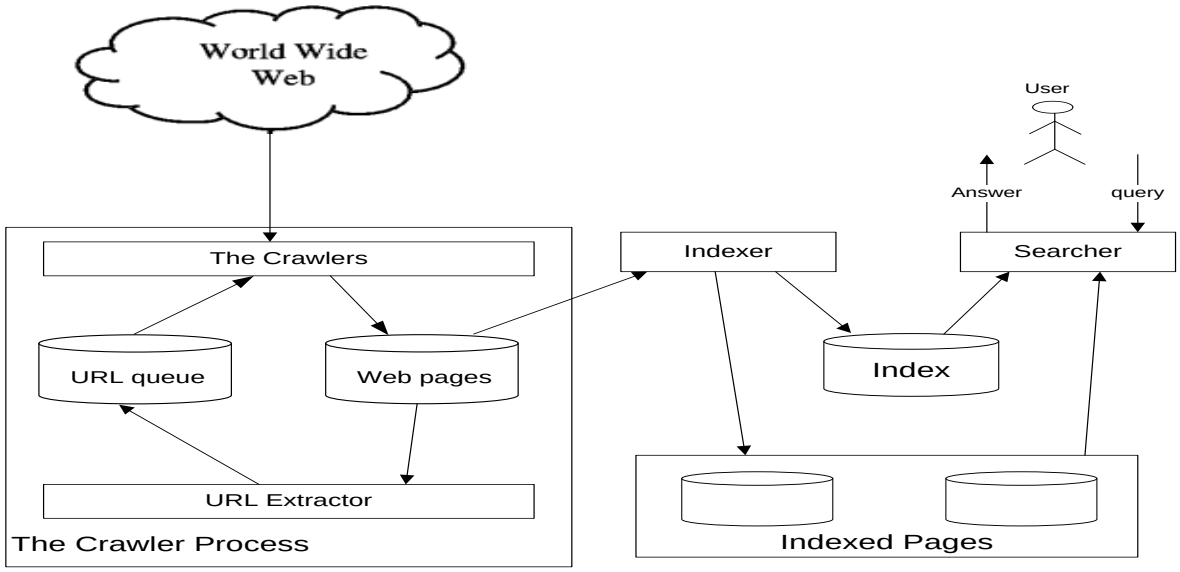

Figure 1: Search Engine Generic Architecture

There are two major classes of indices for text retrieval were proposed: a technique based on inverted lists and superimposed coding scheme proposed in [4] and further enhanced by several authors, e.g. [5]. Both structures have certain scalability problems of different nature. The inverted lists provide good performance for single-keys searches, but their performance rapidly decreases when the query size increases, which is important for text retrieval. The superimposed coding does not depend on the query size, but depends linearly on the data set size, which potentially implies hard scalability problems for huge data collections. So the second technique is hard to the crawled web pages, while the first technique is the more suitable for the crawled web pages which is huge data set in search engines to generate the index data faster.

In last years, many changes exist in the way of perceiving and using computing, which computing needs 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. As 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 [6]. The distributed system exists when there are computers that are interconnected and communicating. The issues in designing, building and deploying distributed computer systems have been discussed in many years. The term of the grid computing was known in the mid 1990s, which refers to a proposed distributed computing infrastructure for advanced science and engineering [8, 9]. The grid computing is a special type of parallel computing which relies on complete computers with onboard CPU, storage, power supply, network interface, etc. these computers connected to a network (private, public or the Internet) by a conventional network interface, such as Ethernet. A grid has a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [10].

Web pages are collections often very large, so it cannot perform index construction efficiently on a single machine. This is particularly true of the World Wide Web content. This required large computer clusters to construct any reasonably sized web index. Distributed indexing algorithms are very important in web search engines for index construction. So, grid computing introduces a good solution to improve the performance through utilizing the available resources. Using indexing algorithm that distributes the process of indexing over a large computer clusters as MapReduce. The point of a cluster solves large computing problems on cheap commodity machines or nodes that are built from standard parts (processor, memory, and disk) as opposed to on a supercomputer with specialized hardware [11]. Grid computing is not only share processing but also share memory between available resources in a network. Index process is large processing and also requires huge memory up to terabyte to store index data, so grid computing will be useful, and it will introduces a solution without needing to expensive computers with huge storage.

The web contents are modified frequently with documents being added, update, or delete. This means that the index needs to update by adding, update, and delete frequently. The simplest way is to periodically reconstruct the index from scratch. This way is a good solution if the number of changes over time is small and a delay in making new documents searchable is acceptable and this requires enough resources are available to construct a new index while the old one is still available for querying [11]. But this way is not suitable for World Wide Web content which frequently updated is made continuously. The best way to make new documents be included quickly in the query results is to use two indexes, main index and the secondary index which the secondary index contains the new document. If any query comes will run on the main index and the secondary index and merges the results to get the newest results up to date.

2. Related work

The performance and the time of information retrieval are affected by a number of factors; the main factors are how this data set is organized which mean indexing and the frequency of change in this data set, so there are a number of research groups that have been working in the field of distributed index and other focused on dynamic indexing. There are researches to study changes and dynamics of the web content. Algorithms and techniques of the indexing are a main point of researches, where there are proposed new algorithms to serve

the problems as the long processing time and the frequency change, also there are a proposed mergence of the existing algorithms to improve the performance. Also there are many research groups that have been working in distributed computing. These groups have created libraries, middleware 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 [7, 8, 9].

Maxim Martynov and Boris Novikov proposed an algorithm for query evaluation in text retrieval systems based on well-known inverted lists augmented with additional data structure and estimate expected performance gains [12]. Justin Zobel, Alistair Moffat, and Ron Sacks-Davis proposed an inverted file indexing scheme based on compression. This scheme allows users to retrieve documents using words occurring in the documents, sequences of adjacent words, and statistical ranking techniques. The compression methods chosen ensure that the storage requirements are small and that dynamic update is straightforward [13]. Eytan Adar, Jaime Teevan, and Susan T. Dumais explores the relationship between Web page content change (obtained from an hourly crawl of over 40K pages) and people's revisitation to those pages (collected via a large scale log analysis of 2.3M users). They identify the relationship, or resonance, between revisitation behavior and the amount and type of changes on those pages [14]. A. Gulli and A. Signorini estimates the size of the public indexable web at 11.5 billion pages. They also estimate the overlap and the index size of Google, MSN, Ask/Teoma and Yahoo! [15]. Martin Klein and Michael L. Nelson present the results from studying the change of titles over time. They take titles from copies provided by the Internet Archive of randomly sampled web pages and show the frequency of change as well as the degree of change in terms of the Levenshtein score [16]. Eytan Adar, Jaime Teevan, Susan T. Dumais, and Jonathan L. Elsas explores changes in Web content by analyzing a crawl of 55,000 Web pages, selected to represent different user visitation patterns. They describe algorithms, analyses, and models for characterizing changes in Web content, focusing on both time and structure [17]. Dionysios Logothetis and Kenneth Yocum explores using indexed data to support stateful groupwise processing. Access to persistent state is a key requirement for incremental processing, allowing operations to incorporate data updates without recomputing from scratch [18]. Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly created a general-purpose distributed execution engine for coarse-grain data-parallel applications this engine called Dryad. Dryad is designed to scale from powerful multi-core single computers, through small clusters of computers, to data centers with thousands of computers. The Dryad execution engine handles all the difficult problems of creating a large distributed, concurrent application: scheduling the use of computers and their CPUs, recovering from communication or computer failures, and transporting data between vertices [19].

3. Alchemi Tool

One of the systems that help in implementing the grid based applications is Alchemi system. This system 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. This system runs on the Windows operating system in the .NET Framework [20]. Alchemi provides an API for C# and C++, and operates in the Windows .NET framework. Alchemi system is an open source software toolkits which is developed at the University of Melbourne. This system 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 2, 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 [21]. 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 [22].

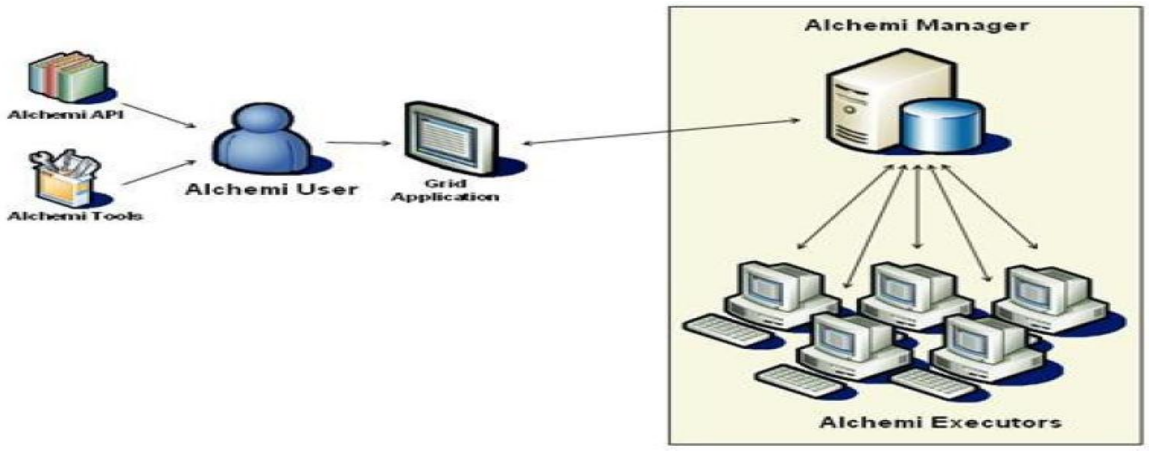

Figure 2: Alchemi's main components

4. Index Architecture

The index operates on the output of the crawler, which the crawler gathers the pages from the World Wide Web and store them in the database. The indexer reads the pages and processes them to generate the index to facility the information retrieval process and reduce the access time when any search query comes. The efficiency of index affects the time required to retrieve the required pages for the user search's queries. This mean that the index technique used affects the searcher performance. The proposed system structure focuses on two main issues: the dynamic indexing process and the distributed indexing process. The first issue is the dynamic indexing that serves changes in Web content where the Web is a dynamic, ever changing collection of information. The second issue is the distributed indexing serves the huge data processing of large collection of web pages which the indexable Web is more than 11.5 billion pages [15]. The dynamic indexing can be achieved by using a secondary index beside the main index. The main index contains the overall index data that have been collected and indexed before receiving any search query. The secondary index is responsible of indexing the pages has been crawled while running the searches where these pages may be new pages or existed but have been updated so the secondary index generates index of these pages and stores them temporarily. When the secondary index reached to the predefined limit, it merges its contents with the main index. When any query comes to the searcher, it sends the query to the main and the secondary index and merges the results of them together to get the result with the latest updates. The figure 3 shows the Top level structure of the search engine with the dynamic index. This structure represented in this figure focuses on the index parts, and its relationship with the others parts of the search engine. The secondary index works while running the system and receiving queries to the searcher. Before receiving any query from searchers while the preparation of the system the main index works alone.

The distributed index achieved by distributing the documents over a set of computers. The proposed architecture to achieve the distributed index consists of a set of computers distributed and one master computer to manage the other computers and distribute the documents over them. Each computer pares the document that has been received from the master and generates the inverted index list of the document then returns the inverted index list back to the master computer; this supports data-parallel indexing. The overall index lists are returned to the master computer, so the memory required to store the index dictionary is very huge memory. The master computer distributes the received lists according to term, where the master computer is connected to other computers in a network with large memory disk and each computer contains an index repository of a set of terms. The master computer sends each subset of the index lists according to terms to a specified computer that contains set of terms, and then this computer merges the received index lists with the existed lists in its database. This architecture distributes the load of processing while parsing by distributing the documents over set of computers, then distributes the memory or the storage by distributing the terms over other set of computers.

Grid computing is a good mechanism to support parallel execution mode using the available resources and minimize the execution cost. So the distributed indexing can be implemented using the environment of the grid computing. The grid computing has a master node that controls the available resources in the grid computing. The crawled documents are passed to the master node of the grid then the master distributes the documents over the available computers in the grid, and receives returned inverted list of each computer. The master is responsible of distributing the returned inverted index according to terms over a number of computers connected to it. Therefore any query comes from the searcher of the search engine to the master computer of the grid; the master computer will send the query to a specified computer to search in it about the query's keyword.

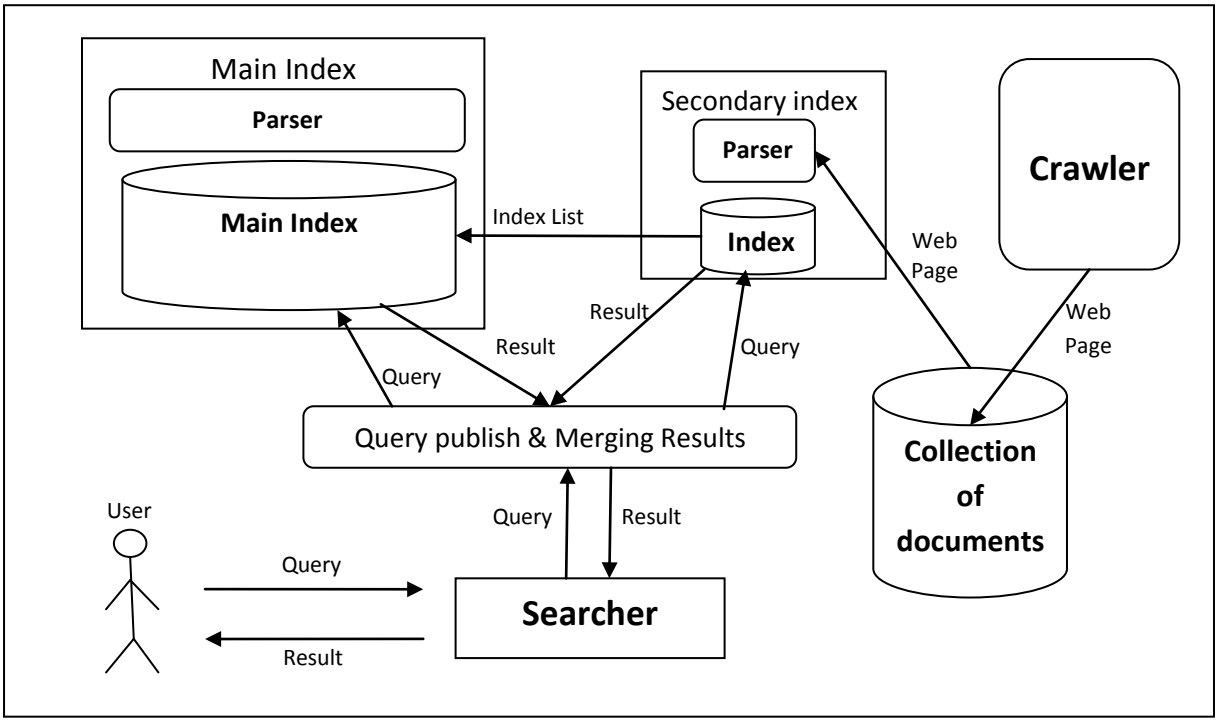

Figure 3: Top level structure of the Dynamic Index

This distribution of index according to terms has a positive impact on the performance of the searcher of the search engine. The figure 4 shows distribution of the parsing load and distribution of the index storage over the computers as mentioned previously.

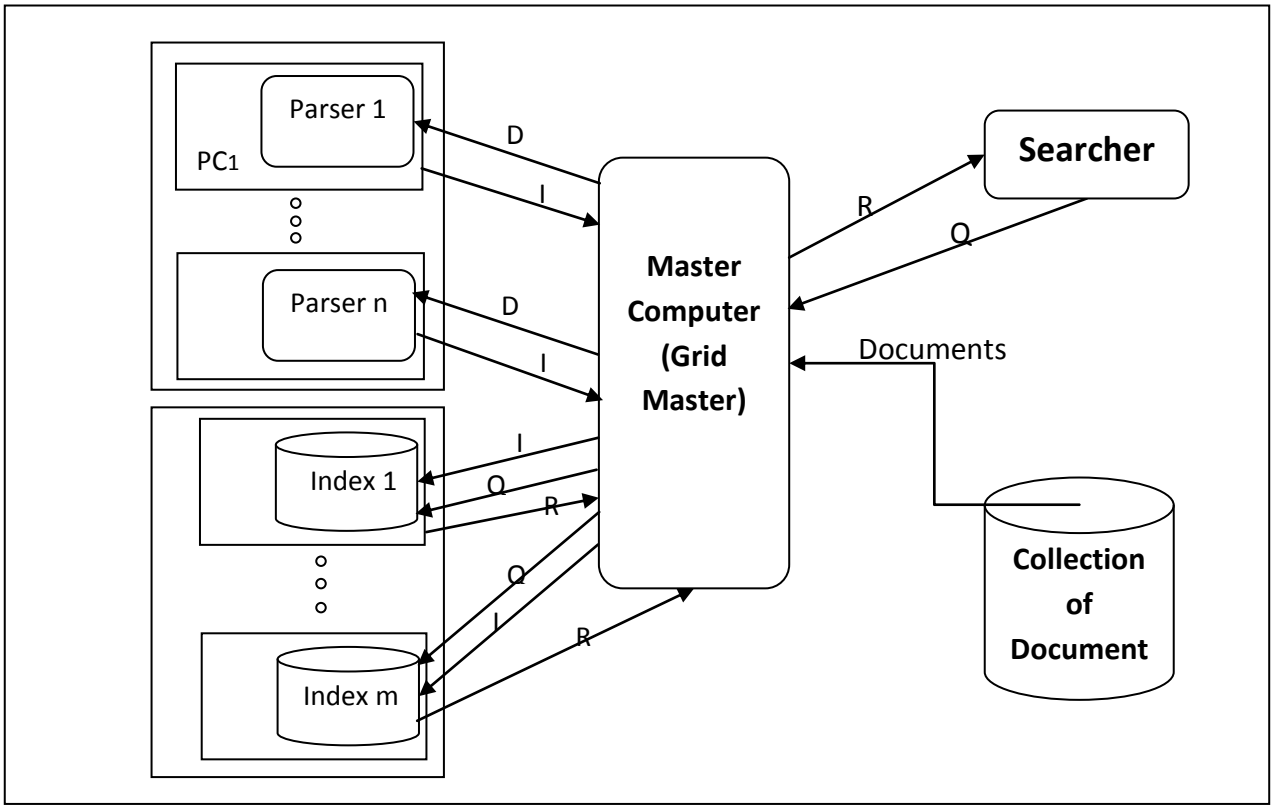

Figure 4: Simple distributed index structure. (R) Result, (Q) Query, (D) Document

It is very useful merging the dynamic index architecture with the distributed index architecture to get the advantages of the dynamic index and the distributed index without needing to resources with a specific capabilities or expensive servers. This can be achieved by using the grid computing to provide available resources in a network. The dynamic index achieved by specifying a computer come its role while running the search engine. This computer read the documents that crawled from web while running the search engine, parses them and generates the inverted index list of these documents. When the index repository reaches to a specified size, the computer send the index content to the master computer of the grid to distribute the index list according to the terms and send them to the computers of the indexes.

The distributed index come its role in the start stage to distribute the load of parsing document over a set of available computers in the grid computing and its role continues to distribute the index according to terms over a set of computers predefined to store the index of all pages that have been crawled and what will comes while the search engine is running. The figure 5 views the architecture of the distributed index and the dynamic index in one architecture design. This architecture shows that there is a computer responsible of publishing the query and merging the returned result. The computer of query publishing and merging results sends the query to the specified computer according to the terms of the query and also sends the query to the computer of the secondary index. The computer of query publishing and merging results receives the results and merges them with a high priority to the result of the secondary index and return the result of merging to the searcher.

5. Implementation

Alchemi system provides a Software Development Kit (SDK) that includes a Dynamic Link Library (DLL) that supports multithreaded applications; it is used to implement the proposed architecture of the index using the grid system.

The architecture is implemented by C# programming language as it is supported by Alchemi. It is designed a master class that play the role of the controller in the architecture, which get the documents from the crawled documents database, distributes the documents over the available computers in the grid, and receives the results of each computer in the grid as a list of keywords in form of a token as shown in the programming model in figure 6. The token is an object of a class contain the keyword, document Id, and a number refers to the frequency of the kerword in the document. Each computer in the grid receives the document from the master, parses the document, groups the words of the document and generates a token for each keyword with the document Identifier and the frequency of the keyword in the document that will help while ranking in the searcher part in the search engine.

The master classifies the tokens according to the first character of the keyword to distribute the index while store according to the keyword. There are five computers with SQL server and contain the database of index each one store a specific set of keywords as the following:

- Computer #1 stores the keywords starts with characters A, B, C, D, E,
- Computer #2 stores the keywords starts with characters F, G, H, I, J,
- Computer #3 stores the keywords starts with characters K, L, M, N, O,
- Computer #4 stores the keywords starts with characters P, Q, R, S, T,
- Computer #5 stores the keywords starts with characters U, V, W, X, Y, Z.

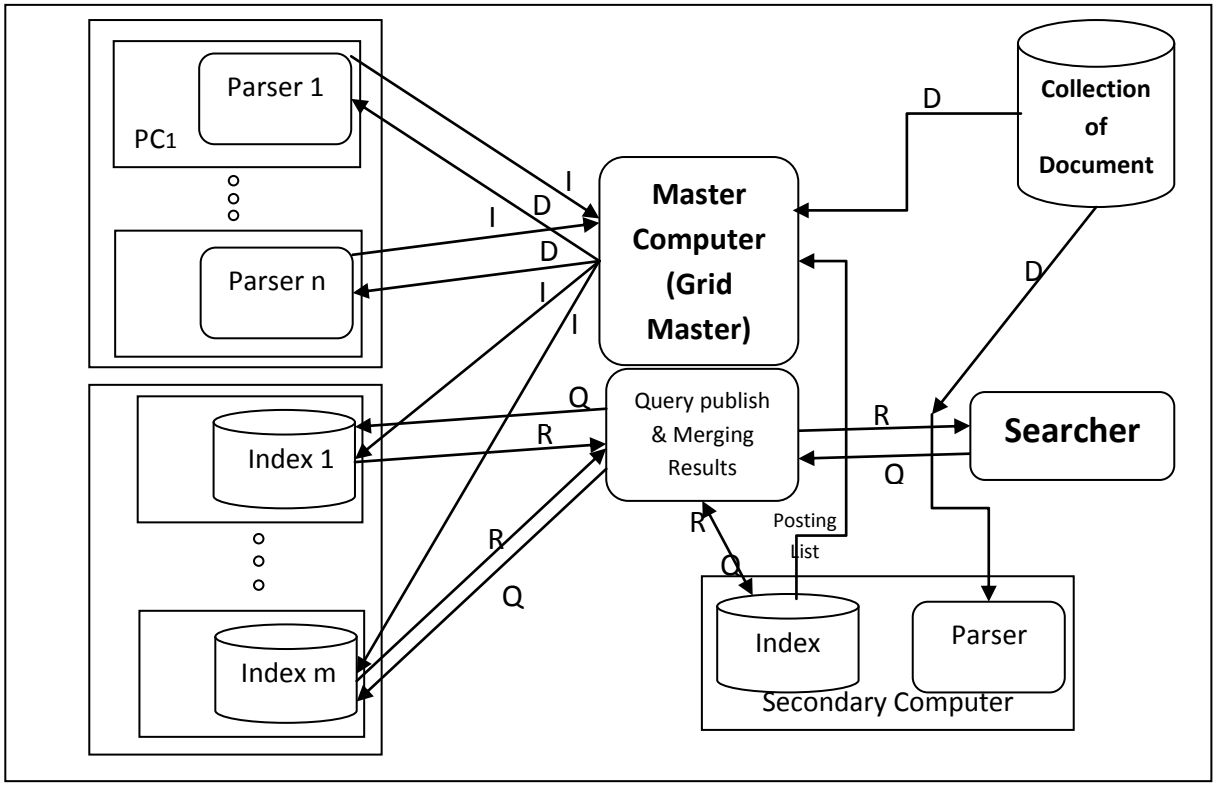

Figure 5: block Diagram of dynamic distributed index (D) Document, (Q) Query, (R) Result

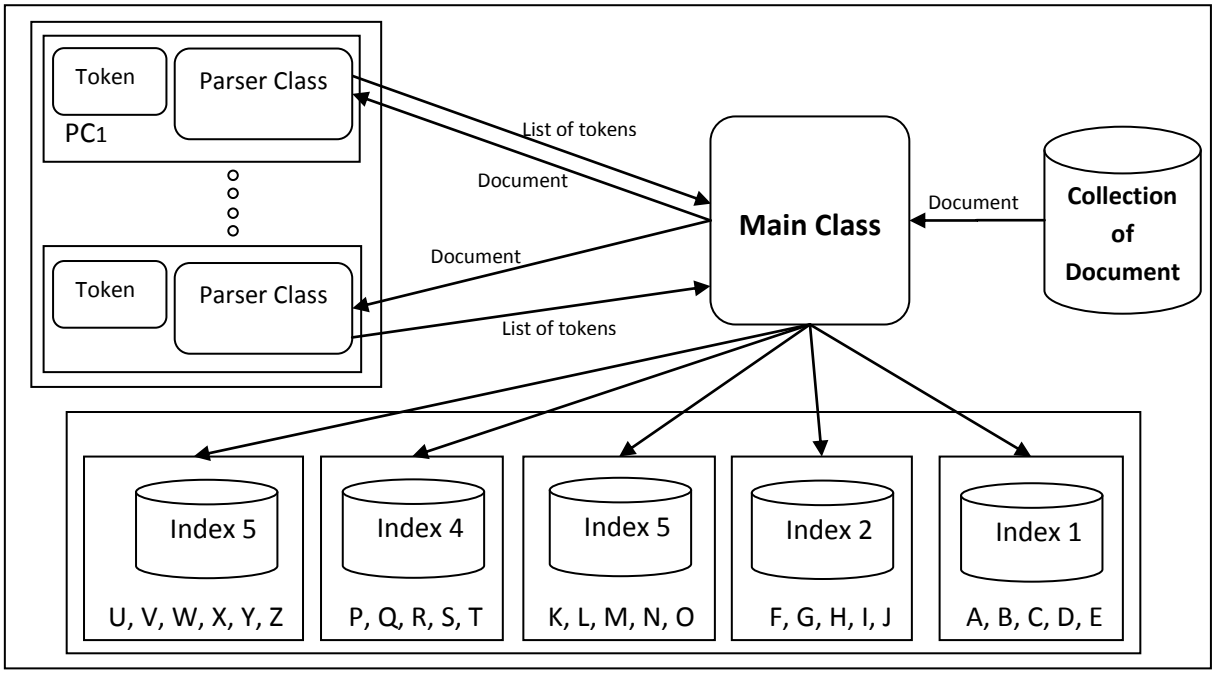

Figure 6: programming module

The programming code of the overall architecture exists in the master computer and the other computers in the grid only contain Alchemi Executor program. It is important to run Alchemi components before running the application. First, run the Alchemi Manager in the master computer as in figure 7, and then run the Achemi Executor in the available computers as in figure 8 to connect the computers to the master computer and schedule the available executors in the grid environment.

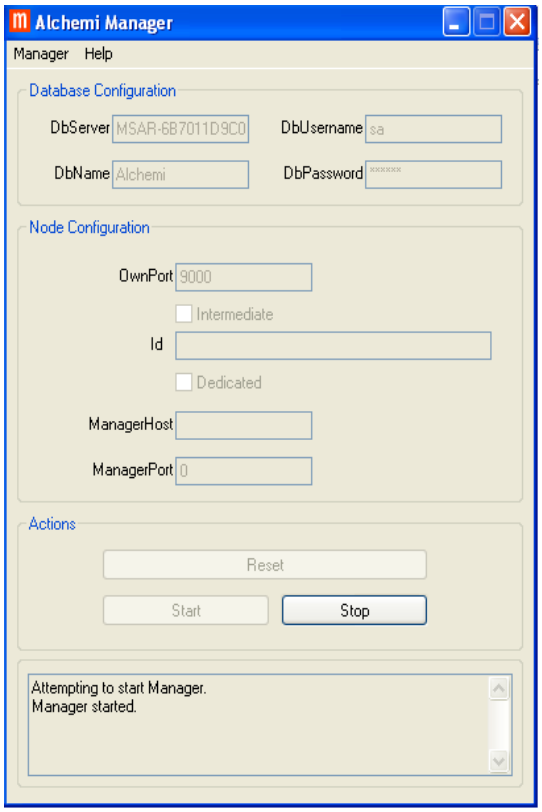

Figure 7: Alchemi Manager Form

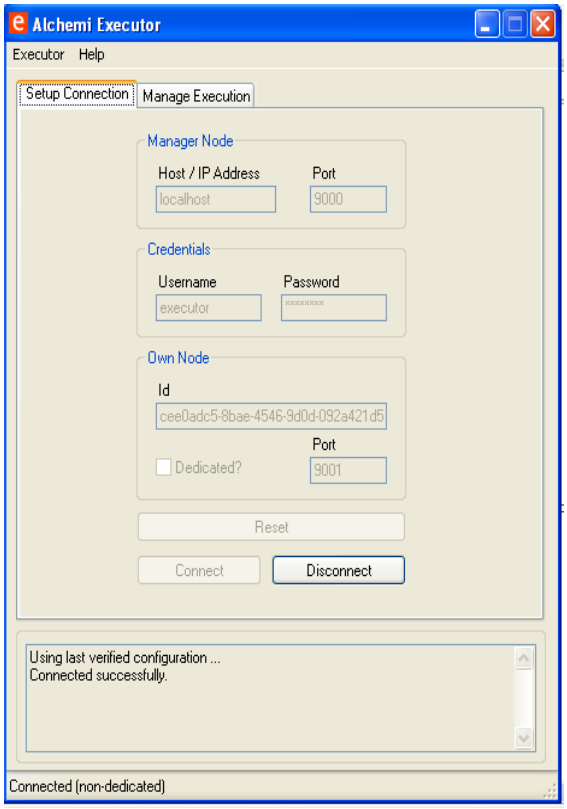

Figure 8: Alchemi Executor Form

6. Results and Evaluation

The used testbed consists of seven personal computers with 1GB RAM and Intel CPU 2.3, Alchemi executer program is installed on five computers as in Figure 8, and Alchemi manager program is installed on only one computer as in Figure 7, where SQL server 2008 is installed. Run the manager computer 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 9, the figure shows the executer table in the Alchemi database which store the details of each executer. When the application runs it generate a set of threads of type GThread. These threads are submitted to the manager to distribute them over the available computers. These threads are stored in the threads table in Alchemi database as in Figure 10.

	executor_id	is_dedicated	is_connected	ping_time	host	port	usr_name	cpu...	cpu_...	cpu_avail	cpu_totalusage
1	B2F88439-61...	1	1	2012-07-21 1...	dell-50cfd542a2	9001	executor	2294	4	54	1321.38
2	98701542-15...	1	1	2012-07-21 1...	dell-12cf3ab235	9001	executor	2294	96	96	464.72
3	A6F1C95E-F5...	1	1	2012-07-10 1...	dell-83b7fa8b24	9001	executor	2294	0	94	2.2
4	7CC22710-CC...	1	1	2012-07-21 1...	dell-92c6fab222	9001	executor	2294	1	1	760.579999999996
5	841D0C66-A8...	1	1	2012-07-21 1...	dell-654ac2f541	9001	executor	2294	0	0	2569.08000000001
6	D2417D6B-6...	1	1	2012-07-21 1...	dell-321ff52410	9001	executor	2294	0	86	981.700000000003
7	9084E6BA-8A...	1	1	2012-07-21 0...	dell-4758a6bc14	9001	executor	2295	4	31	703.719999999999

Figure 9: Achemi Executors connected to manager

	internal_thread...	application_id	executor_id	thread_id	state	time_started	time_finished	priority
▶	373601	c61d2636-d408-...	7cc22710-cc3d-...	0	4	2012-07-21 16:...	2012-07-21 16:...	5
	373602	c61d2636-d408-...	841d0c66-a802-...	1	4	2012-07-21 16:...	2012-07-21 16:...	5
	373603	c61d2636-d408-...	841d0c66-a802-...	2	4	2012-07-21 16:...	2012-07-21 16:...	5
	373604	c61d2636-d408-...	7cc22710-cc3d-...	3	4	2012-07-21 16:...	2012-07-21 16:...	5
	373605	c61d2636-d408-...	841d0c66-a802-...	4	4	2012-07-21 16:...	2012-07-21 16:...	5
	373606	c61d2636-d408-...	7cc22710-cc3d-...	5	4	2012-07-21 16:...	2012-07-21 16:...	5
	373607	c61d2636-d408-...	841d0c66-a802-...	6	4	2012-07-21 16:...	2012-07-21 16:...	5
	373608	c61d2636-d408-...	7cc22710-cc3d-...	7	4	2012-07-21 16:...	2012-07-21 16:...	5

Figure 10: The Threads table in the alchemi database

If only one computer is available, and the indexing occurred sequentially then the total time is

$$T = X * (t_P + t_m)$$

X Be number of documents,

t_P Time of parsing one document,

t_m Time of merging the document keywords with the other index

But if (N) nodes are available, then total time is

$$T = \frac{X * t_P}{N} + (X * t_m)$$

N Be the number of executor computers.

Thus, by increasing the number of Executor computers, the indexing time will decrease, proportionally to the number of executor nodes in the Grid. This is evident in the results that shown in Table 1 and the graph in Figure 11.

The results of running the application number of times with different number of executors up to five executors, and different numbers of documents in each experiment. The numbers of the executors, the numbers of the documents that run on the application and the results of running the application in seconds are stored in the table 1. The different experiments show that there is a clear reduction and improvement in the execution time of the indexing process using distributed indexing technique as shown in figure 11.

Table 1. Table captions should be placed above the table

no. of Pages	50	100	200	300	400	500
2 Executor	41	83	168	249	337	417
2 Executor	18	35	71	108	142	179
3 Executor	14	29	59	88	119	146
4 Executor	12	24	49	74	99	124
5 Executor	10	20	41	61	81	103

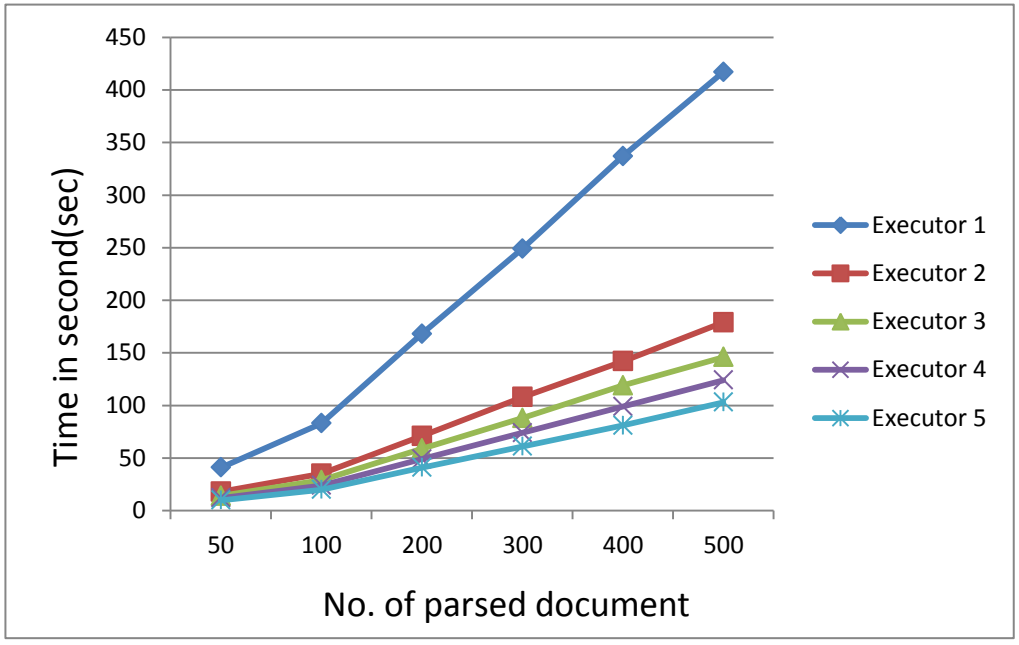

Figure 11: Execution time using varying number of executors enabled

This evaluation of distributed indexing occurred through calculating of the execution time and view the time reduction by distributing the parsing process but there is also another benefits of this architecture that will affect the performance of the search process by distribute the index through the key words instead of searching in all index, search in a subset of the index based on the first character of the keyword of the query.

The performance of the dynamic index tested by make some changes in set of documents in the crawled documents in the database then run a search query on the index database after two minutes, the returned documents affects by retrieve the new version of documents that have been changed in the crawled pages in the database.

7. Conclusion and Future Work

In this paper, an architecture of distributed index proposed using grid computing. This architecture have the advantages of distributed index according to document while parsing the document to extract the terms form document and the advantages of distributed index according to terms while storing the index and utilizing the available resources in a grid without needing to a huge storage. Distribution while parsing the document to extract terms achieved using grid computing technique. This architecture eliminates the needs to computers with high speed processing such as supercomputer and also eliminates the needs to computers with huge storage in one place. Also this architecture give a better performance while searching instead of searching in all data indexed the master node will identify smaller scope of data based on the query keyword. The paper maintained the dynamic index in the proposed architecture by specifying a computer continuously to index the updates in the crawled document and transfer the index to the master node to distribute it.

In future work, it is planned to look into dynamic load balancing algorithm, optimizing the performance of the searching, and using grid computing to enhance the performance of the other search engine components such as Page Ranking and building a complete search engine components based on the grid computing.

8. REFERENCES

[1] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", Computer Networks and ISDN Systems, 30(1–7):107–117, April 1998.

[2] Clarke, C., Cormack, and G., "Dynamic Inverted Indexes for a Distributed Full-Text Retrieval System", TechRep MT-95-01, University of Waterloo, February 1995.

[3] G. Huck, F. Moser, and Erich J. Neuhold, "Integration and handling of hypermedia information as a challenge for multimedia and federated database systems", In Proc. Of the Second Intnl. Workshop on Advances in Databases and Information Systems - ADBIS'95, pages 183–194, Moscow, June 27–30 1995.

[4] C. Faloutsos and S. Christodoulakis, "Signature files: an access method for documents and its analytical performance evaluation". ACM Trans. on Database Systems, 4(2):267–288, 1984.

[5] A. Kent, R. Sacks-Davies, and K. Ramamohanarao, "A superimposed coding scheme based on multiple block descriptor files for indexing very large databases". In Proc. 14 conf. VLDB, pages 351–359, 1988.

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

[7] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations", International Journal of High Performance Computing Applications Fall 2001 15: 200-222.

[8] I. Foster, C. Kesselman, J. M. Nick, S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Global Grid Forum, June 22, 2002.

[9] F. Berman, G. Fox, and T. Hey, Book: "Grid Computing: Making the Global Infrastructure a Reality", published March 2003.

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

[11] C. D. Manning, P. Raghavan, and H. Schutze, Book: "Introduction to Information Retrieval", Cambridge University Press 2008.

[12] M. Martynov and B. Novikov, "An Indexing Algorithm for Text Retrieval", Proceedings of the International Workshop on Advances in Databases and Information Systems (ADBIS'96). Moscow, September 10–13, 1996.

[13] J. Zobel, A. Moffat, and R. Sacks-Davis, "An Efficient Indexing Technique for Full-Text Database Systems", Proceedings of the 18th VLDB Conference Vancouver, British Columbia, Canada 1992.

[14] E. Adar, J. Teevan, and S. T. Dumais, "Resonance on the Web: Web Dynamics and Revisitation Patterns", ACM 978-1-60558-246-7/08/04, Boston, MA, USA, April 4–9, 2009.

[15] A. Gulli and A. Signorini, "The Indexable Web is More than 11.5 billion pages", ACM 1595930515/05/0005, Chiba, Japan , May 10–14, 2005.

[16] M. Klein and M. L. Nelson. "Investigating the Change of Web Pages' Titles Over Time", InDP'09, Austin, TX, USA, June 19, 2009.

[17] E. Adar, J. Teevan, S. T. Dumais, and J. L. Elsas, "The Web Changes Everything: Understanding the Dynamics of Web Content", ACM 978-1-60558-390-7, WSDM'09, Barcelona, Spain, February 9-12, 2009.

[18] D. Logothetis and K. Yocum, "Data Indexing for Stateful, Large-scale Data Processing", ACM, NetDB '09 Big Sky, MT USA, 2009.

[19] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks", ACM 978-1-59593-636-3/07/0003, Lisboa, Portugal, March 21–23, 2007.

[20] A. 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.

[21] A. Luther, R. Buyya, R. Ranjan and S. 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.

[22] K. Nadiminti, Yi-Feng Chiu, N. Teoh, A. Luther, S. Venugopal, and R. 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.