Analyzing the Load Balance of Term-based Partitioning

Ahmad Abusukhon
Faculty of Science & IT
Al-Zaytoonah Private University of Jordan
Amman Jordan
ce4aab@student.sunderland.ac.uk

Mohammad Talib
Department of Computer Science
University of Botswana
Private Bag UB 00704, Gaborone, BOTSWANA
talib@mopipi.ub.bw

Abstract— **In parallel (IR) systems, where a large-scale collection is indexed and searched, the query response time is limited by the time of the slowest node in the system. Thus distributing the load equally across the nodes is very important issue. Mainly there are two methods for collection indexing, namely document-based and term-based indexing. In term-based partitioning, the terms of the global index of a large-scale data collection are distributed or partitioned equally among nodes, and then a given query is divided into sub-queries and each sub-query is then directed to the relevant node. This provides high query throughput and concurrency but poor parallelism and load balance. In this paper, we introduce new methods for terms partitioning and then we compare the results from our methods with the results from the previous work with respect to load balance and query response time.**

Keywords- ***Term-partitioning schemes, Term-frequency partitioning, Term-lengthpartitioning, Node utilization, Load balance***

I. Introduction

The number of pages (documents) available online is increasing rapidly. Gulli and Signorini [17] estimated the current size of the web. They mentioned that Google claims to index more than 8 billion pages. They estimated the indexable web to be at least 11.5 billion pages. Beside the huge document collection, we have a large number of information requests (queries) that are submitted by clients. Sullivan [18] reported that the number of searches per day performed by Google is 250 million. In order to the users to effectively retrieve documents that are relevant to their needs, the IR systems must provide effective, efficient, and concurrent access to large document collections. Thus, the first step in developing information retrieval system is to decide on what access method should be used in order to access large-scale collection efficiently. In IR systems the indices of documents must be built to perform timely information retrieval. The most known structures for building the index of large-scale collection are inverted files and signature files. The most common and most efficient structure for building the index of large-scale collection is the inverted file [1,2].

Zobel[3] compared inverted files and signature files with respect to query responsetime and space requirements. They found that inverted files evaluate queries in less time than signature files and need less space, thus for efficiency reasons, we use the inverted files in our research.

In general, inverted files consist of vocabulary and a set of inverted lists. The vocabulary contains all unique terms in the whole data collection; while the inverted lists composed of a list of pointers and each pointer consists of document identifier and term frequency. The term frequency in each pair represents how many times term i appears in document j (Fi,j). Let's suppose that the inverted list for term "world" is:

World 2:5, 6:3, 12:1, 15:1

This means that term world appears five times in document 2, three times in document 6, one time in document 12, and one time in document 15. The numbers 2,6, 12, and 15 are called the document identifiers while the numbers 5, 3, 1, and 1 are called the term frequencies.

In parallel IR system when term partitioning scheme is used all unique terms in the data collection and their inverted lists reside on a single node called the broker. The broker distributes all terms and their inverted lists across nodes using different approaches. The terms, for instance, may be distributed in round robin fashion. In this case the broker iterates over all terms in the inverted file, and distributes them sequentially across nodes. The aim of round robin partitioning scheme is to balance the load over the nodes by storing nearly equal number of terms on all nodes.

Moffat[4] showed that distributing the terms of the term-based index in round robin fashion results in load imbalance especially when there are heavy loaded terms. Because of this the round robin partitioning scheme does not take care of those terms to be distributed equally across nodes.

Xi[5] proposed the hybrid partitioning scheme in order to achieve load balance. In hybrid partitioning scheme the inverted lists of the term-based index are split into chunks then chunks are distributed across nodes. They investigated partitioning the inverted list into different sizes and they concluded that hybrid partitioning scheme achieves better load balance than the other schemes (document-based and term-based partitioning) when the chunk size is small (1024 posting). But when the chunk size is large the hybrid partitioning is worse than the document partitioning. In this paper, we propose two methods for term partitioning scheme - term length partitioning and term frequency partitioning.

Abusukhon et al. [13, 14, 16] proposed improving the load balance of hybrid partitioning using hybrid queries. In their work, they divided the nodes into clusters then the inverted lists of all terms were divided into a number of chunks, the chunks of a given term that start with a certain letter were distributed equally among the nodes of a certain cluster. A hybrid query was generated from a set of queries and then this query was divided into streams with respect to the first letter of each term. Each stream was directed to the relevant cluster.

II. Related Work

Inverted files can be partitioned by different approaches. Different approaches of data partitioning leads to different load balance and different query response time as described by Abusukhon et al. [15]. In this section we shed light on various strategies for term-partitioning schemes as described in the previous work.

Cambazoglu[6] demonstrated two main types for inverted file partitioning - term-based partitioning and document-based partitioning. In term-based partitioning all unique terms in the data collection and their inverted lists reside on a single node. In document-based partitioning the data collection is divided into sub-collections, sub-collections are distributed across nodes, and then each node builds its own index.

Jeong [7] proposed two methods for load balancing when using term-based partitioning scheme. In the first method they proposed to split the inverted list into equal parts and then distribute those parts across nodes instead of distributing equal number of terms across nodes in order to achieve better load balance.

In the second method they proposed to partition the inverted lists based on the access frequency of terms in the user query and the inverted list size for each term appears in the query. They studied the performance of the above schemes by simulation under different workloads.

Marin and Costa [19] stated that load balance is sensitive to queries that include high frequency terms that refer to inverted lists of different sizes.

Moffat[4] examined different methods to balance the load for term-distributed parallel architecture and proposed different techniques in order to reduce the net querying costs. They defined the workload as follows:

$$Wt = Qt * St$$

Where Wt is the workload caused by the term t that appears in a query batch Qt and has an inverted list of length equals St bytes. The workload for a given node is the sum of Wt of the terms distributed over that node. In one of their experiments the terms of the queries were distributed over the nodes randomly. The simulation result showed that some nodes were heavy-loaded because they retrieved very large size inverted lists; therefore, some of the nodes in the system were half-idle and affect the system throughput. In order to improve the load balance they proposed distributing the inverted lists equally among the nodes based on the number of pointers P in each inverted list.

Jeong and Omiecinski [20] concluded that partitioning by term resulted in load imbalance because some terms were more frequently requested in a query. Thus, nodes where these terms associated with their inverted lists were stored would be heavily utilized.

Xi[5] proposed a hybrid partitioning scheme in order to distribute terms across the nodes. Hybrid partitioning scheme avoids storing terms with long posting lists on one node instead of the inverted list of a given term is split into a number of equal size chunks and then distributed randomly across the nodes. They measured the load balance and concluded that the hybrid-partitioning scheme outperforms other schemes when the chunk size is small. In this paper, we propose Term Length partitioning and Term Frequency partitioning for improving the load balance of term-based partitioning.

III. System Architecture

Fig.1 shows our system architecture. It consists of six nodes and one broker. All nodes are connected to the broker via Ethernet switch.

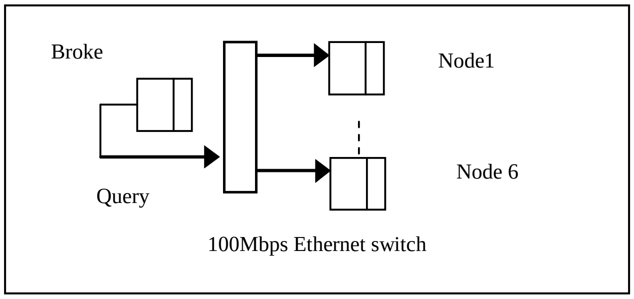

Figure 1. Distributed IR Architecture

The machine specifications for five nodes are: CPU 2.80Ghz RAM 256MB whereas the specification for the last and the broker are: CPU 3.00Ghz, RAM 512MB. All machines are running in Windows XP environment.

IV. Research Methodology

We carried-out a set of real experiments using six nodes and one broker as shown in Fig. 1. In all of our experiments we use the data collection WT10G from TREC-9 and 10,000 queries extracted from the start of Excite-97 log file. The chronology of our research methodology is traced below:

1. We build the global index (called the term-based partitioning) in the following way:

a. Broker sends the documents across nodes in round robin fashion.

b. Each node when receiving its document performs these activities-

• Filters the document it receives from stop words (the stop word list consists of 30 words), HTML tags, and all noncharacters and non-digit terms.

• Accumulates the posing lists in main memory until a given memory threshold is reached. At this point the data stored in memory is flushed to on-disk file [8,9, 2,11]. This process is repeated until all documents in the data collection are indexed.

• Merge all on-disk files together into one on-disk file called the local index or the document-based partitioning.

c. Finally, the broker collects all local indices from all nodes and merges them together in order to produce the global index.

2. We partition the terms of the global index across nodes using four different approaches, viz., round robin partitioning, partitioning based on the length of the inverted list, term length partitioning and term frequency partitioning.

Next, we demonstrate the above approaches and then run a set of real experiments in order to compare them with respect to the node utilization.

A. Round Robin Partitioning

In round robin partitioning, we distribute the terms of the global index across nodes. If we have three nodes and four terms A, B, C, and D associated with their posting lists then term A may reside on node 1, term B on node 2, term C on node 3, and term D on node 1, and so on [10].

B. Term Partitioning Based on the Length of the Inverted List

In this method of partitioning, we pass over the terms of the global index twice. In the first pass, we calculate the length of the inverted list L for each term T, store T and L in a look up file PL after sorting them on L in ascending order. In the second pass, we distribute the terms and the inverted lists of the global index across the nodes using the PL in round robin fashion in the follow order:

1. Read one record (L, T) from PL

2. Search T in the global index and retrieve its inverted list

3. Send T and its inverted list to a certain node in round robin fashion.

4. If no more records then EXIT else go to step1.

We use the above algorithm in order to guarantee that all inverted lists of the same length reside on all nodes equally. Fig. 2 shows an example of the PL file.

Inverted List Length	Term
100	a
100	b
.	.
.	.
1200	z

Figure 2. Sorted look up file (PL)

C. Term Length Partitioning

Case[12] described Zipf's principle of least effort. He stated that:

"According to Zipf's law (1949) each individual will adopt a course of action that will involve the expenditure of the probable least average of his work, in other words, the least efforts"

He wrote that the statistical distribution of words in the text of James Joyce's Ulysses follows the type of pattern on which Zipf based his theory. The 10th most common word appears 2,653 times; the 100th most common word, 265 times; and the 1,000th, 26 times. This relation is called "harmonic distribution". He stated that humans try to use short, common words whenever they can rather than longer words that take more effort. This is the first motivation for the term-length partitioning. In this section, we propose to partition the terms of the global index associated with their inverted lists with respect to the term length (in letters).

Our research hypothesis for term length partitioning is based on statistical information collected from the query log file Excite-97. This information is stored into a look up file as it is shown in Fig.2. In Fig. 3, we see that the term lengths are not distributed equally in Excite-97 (i.e. have very skewed distribution). For example, the number of terms of length 5 equals 360093 while the number of terms of length 11 equals 59927. The total number of terms in Excite-97 is 2235620. Thus the percentage of the terms of length 5 to the total number of terms =360093 / 2235620 = 0.16% while the percentage of terms of length 11 = 59927 / 2235620 = 0.03%. This is the second motivation for the term-length partitioning. When users submitted their queries, the queries contain terms of different length. Suppose that the majority of those terms are of length 4 and 5 as it is shown in Fig. 3. In addition, we partitioned the terms of the global index in round robin fashion and all terms of length 4 and 5 resided on one or two nodes. This way of partitioning will result in load imbalance because most of the work will be carried out by one or two nodes only while other nodes doing less work or may be idle. Thus our hypothesis is that all terms of the same length must be distributed equally a cross nodes in order to achieve more load balance.

Our partitioning method requires passing over all terms of the global index twice. In the first pass, we calculate the length WL of each term T, store WL and T in a look up file PL after sorting them on W_L in ascending order. In the second pass, we distribute the terms and the inverted lists of the global index across nodes using the PL in round robin fashion in the following order:

1. Read one record (WL, T) from PL
2. Search T in the global index and retrieve its inverted list
3. Send T and its inverted list to a certain node in round robin fashion.
4. If no more records then EXIT else go to step1.

We use the above algorithm in order to guarantee that all terms of the same length reside on all nodes equally.

Our proposed partitioning algorithm differ from round robin partitioning in that it distributes the terms across the nodes equally, and it also guarantee that all nodes get the same number of terms of the same length. The round robin algorithm distributes the terms across the nodes equally regardless the term length. Therefore, we expect that the term length-partitioning scheme achieves better load balance than the round robin partitioning. To the best of our knowledge, no previous work investigated partitioning the global index based on the term length, or measured the nodes utilization when using the term length partitioning.

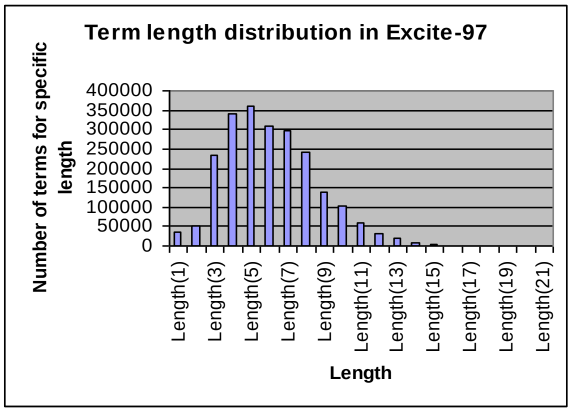

Figure 3. Term length distribution for Excite-97

D. *Term Frequency Partitioning*

Baeza[1] demonstrated Zipf's law (Fig.4), which is used to capture the distribution of i-th most frequent word is $1/i^r$ times that the most frequent word" (r between 1.5 and 2.0), thus the frequency of any word is inversely proportional to its rank (i.e. i-th position) in the frequency table. They showed that the distribution of sorted frequencies (decreasing order) is very skewed (i.e. there were a few hundred words which take up 50% of the text) thus words that are too frequent like stop words can be ignored.

In Fig. 4, graph (A) shows the skewed distribution of the sorted frequencies while graph (B) is the same as graph (A) but we divided the curve into six clusters (A, B, C, D, E, F) after ignoring the stop words. Cluster (A) has the most frequent terms, then cluster B, then C, and so on. In addition, in graph (B) we assume that most or all of the query terms appear in cluster (A), that all or most of the terms in cluster (A) may not reside on all nodes but on one or two nodes in the system. In this case, we may have one or two nodes busy answering the query terms while other nodes are idle, and thus cause the load imbalance.

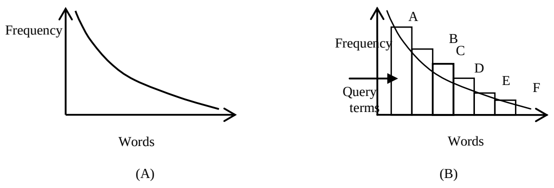

Figure 4. Zipf's Law

Our hypothesis is that if we filter the data collection from stop words (words like the, in, on, …, etc), then the terms with high total frequency (for example the terms in cluster A) are more likely to appear in the user query (i.e. have higher probability to appear in the user query) than the terms with low total frequency. Thus, the terms with high frequency must be distributed equally across the nodes in order to achieve more load balance. Here we propose to partition the terms of the global index with respect to the total term frequency calculated from their inverted lists. To make it clear what we mean by term frequency, we demonstrate the following example: Let's suppose we have two terms (A, B) associated with their inverted lists as it is shown in table 1.

TABLE I. INVERTED LISTS

term	Inverted list
A	1:3, 4:1, 6:2, 9:5, 12:5, 13:2, 15:3
B	2:1, 4:1, 7:2, 9:2, 11:1, 12:1

Then, the total frequency F for each term is calculated as follows:

$F_A = 3+1+2+5+5+2+3 = 21$

$F_B = 1+1+2+2+1+1= 8$

Based on the above calculations, the total frequency of term A is higher than the total frequency of term B and thus we expect that term A has higher probability to appear in the user query than term B. We expect that the load imbalance may occur, if the majority of the terms with higher total frequency reside on one or two nodes, as a result of performing some techniques like round robin partitioning, in this case, most of the user query terms are answered by one or two nodes and thus cause the load imbalance. In the next section, we show how to calculate the probability of a given term to appear in the user query terms.

1) Calculate the Term Probability

Suppose that we have the document collection Ω where:

$$\Omega = \{D_0, D_1, D_2, \ldots, D_n\}$$

Let $T = \{t_0, t_1, t_2, \ldots, t_n\}$ be the set of terms appears in any document D_i in any combination. Let the term t_j occurs m times in D_i, then we assume that the probability (Pt_{ji}) that the term t_j appears in the query terms is equivalent to how many times it occurs in the whole data collection.

$$p_{tj} = \sum_{i=1}^{n} m_{j,i} \quad (1)$$

Where, n is the total number of documents in the data collection and $m_{j,i}$ is how many times the term j appears in document i.

For example, suppose we have a data collection contains 4 documents (d1, d2, d3, and d4) and three terms (t1, t2, and t3) and that t1 appears in these documents (5, 10, 2, 3) times, t2 appears (1, 3, 0, 1) and t3 appears (1, 1, 1, 2). We assume that the probability of term t1 to appear in the query terms is 20, t2 is 5 and t3 is 5.

To normalize the value of Ptj, we divided it by the summation of the total frequencies of all distinct terms in the data collection, i.e.

$$p_{tj} = \frac{\sum_{i=1}^{n} m_{j,i}}{\sum_{k=1}^{n}\sum_{l=1}^{s} m_{l,k}} \quad (2)$$

Where, n is the total number of documents in the data collection and s is total number of distinct terms in the data collection. In the above example, after normalization, the probability of term t1 = 20 / 30 (i.e. 0.7) while the probability of term t2 = 5/30 (i.e. 0.17).

2) Term Distribution Based on the Total Term Frequency

We pass over all terms of the global index twice. In the first pass, we calculate the total frequency F for each term T using equation 1 then store F and T in a look up file PL after sorting them on F in ascending order.

In the second pass, we distribute the terms and the inverted lists of the global index using the PL in round robin fashion in the following order:

1. Read one record (F, T) from PL
2. Search T in the global index and retrieve its inverted list
3. Send T and its inverted list to a certain node in round robin fashion
4. If no more records then EXIT else go to step1.

The above algorithm is used in order to guarantee that all terms of the same total frequency F are distributed across all nodes equally.

To the best of our knowledge, no previous work investigated partitioning the global index based on the total term frequency or measured the nodes utilization when using the term frequency partitioning.

V. EXPERIMENTS

In this research, we carried out a set of real experiments using the system architecture shown in Fig. 1. We used the data collection WT10G from TREC-9 in order to build the global index and 10,000 queries extracted from the start of the Excite-97 log file to measure the node utilization for each node.

Xi[5] defined the node utilization as – "the total amount of time the node is serving requests from the IR server divided by the total amount of time of the entire experiment".

We distributed the terms of the global index using the four approaches mentioned in sections A, B, C, and D. We carried out 10,000 queries, each query is sent across all nodes. Each node retrieves and sends the inverted lists of the query terms to the broker for evaluation. We considered the time the node serving the query S_t to be the time required to retrieve all inverted lists of query terms and send them to the broker. We considered the node to be idle if the query term does not exist on its hard disk in that case, the searching time is excluded from S_t. The total time for each experiment is shown in table VII. For each partitioning scheme mentioned in sections A, B, C, and D, we calculated ΔU:

$$\Delta U = \text{Maximum node utilization} - \text{Minimum node utilization} = \text{MaxU} - \text{MinU} \quad (3)$$

Tables II, III, IV, and V show the time taken by each node to serve 10,000 queries sent by the broker as well as the node utilization. We calculated the node utilization by dividing the time the node serving queries by the total time of the experiment. For example, the node utilization for node 1 (Table II) is calculated as follows:

Node utilization = 598195 / 3349515 = 0.1785.

This calculation step is carried out for all tables (II, III, IV, and V). Next, we produce table VI from the above tables. For each table we got the minimum and the maximum node utilization (MinU, MaxU). For table II, MinU = 0.1104 and MaxU= 0.1947 then we calculate ΔU:

$$\Delta U = MaxU - MinU$$
$$= 0.1947 - 0.1104$$
$$= 0.0843$$

We use table VI to produce Fig. 5 and we calculate the average query response time for each of the above partitioning algorithms by dividing the total time of the experiment by the total number of the executed queries. For round robin partitioning scheme:

The average query response time = 3349515 / 10000

= 334.95 milliseconds

Table VIII and Fig. 6, show the partitioning methods and the average query response time.

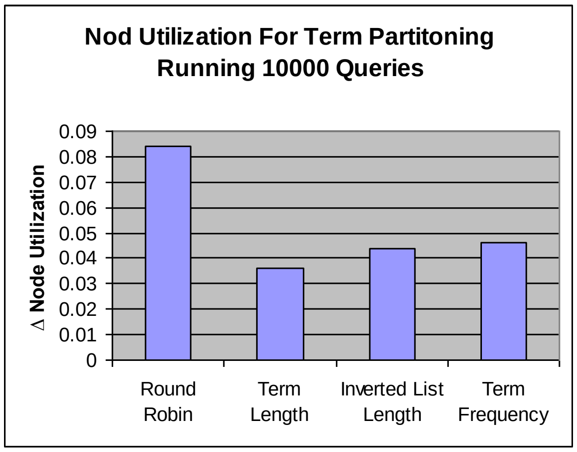

Figure 5. Comparison between four approaches for term partitioning scheme (Round robin, Term Length, inverted List Length, and Term frequency) with respect to ΔU

TABLE II. NODE UTILIZATION FOR ROUND ROBIN PARTITIONING

Node #	Time serving queries (milliseconds)	Node utilization
1	598195	0.1785
2	541421	0.1616
3	369798	0.1104
4	652215	0.1947
5	628682	0.1876
6	604870	0.1805

TABLE III. NODE UTILIZATION FOR PARTITIONING BASED ON THE LENGTH OF INVERTED LIST.

Node #	Time serving queries (milliseconds)	Node utilization
1	667148	0.2046
2	596106	0.1828
3	640117	0.1963
4	699275	0.2145
5	647914	0.1987
6	581225	0.1782

TABLE IV. NODE UTILIZATION FOR PARTITIONING BASED ON THE TOTAL TERM FREQUENCY

Node #	Time serving queries (milliseconds)	Node utilization
1	559151	0.1692
2	640726	0.1939
3	590804	0.1788
4	594265	0.1799
5	667375	0.202
6	711796	0.2154

TABLE V. NODE UTILIZATION FOR PARTITIONING BASED ON TERM LENGTH

Node #	Time serving queries (milliseconds)	Node utilization
1	596510	0.1690
2	535703	0.1518
3	689623	0.1954
4	625451	0.1772
5	629086	0.1783
6	618969	0.1754

TABLE VI. Δ NODE UTILIZATION

Term Partitioning Scheme	Δ Node Utilization (Max - Min)
Round Robin	0.0843
Term Length	0.0363
Inverted List Length	0.0436
Term Frequency	0.0462

TABLE VII. TOTAL TIME OF EXPERIMENTS

Term partitioning method	Total time of experiment (milliseconds)
Round Robin	3349515
Length of inverted list	3259879
Term frequency	3303175
Term length	3528047

TABLE VIII. AVERAGE QUERY RESPONSE TIME (MILLISECONDS)

Partitioning Method	Average Query Response Time (milliseconds)
Length of inverted list	325.9879

Round Robin	334.9515
Term frequency	330.3175
Term length	352.8047

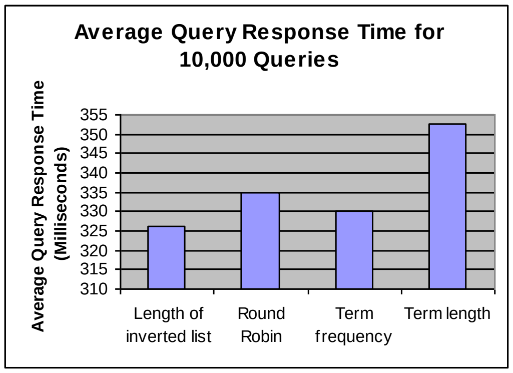

Figure 6. Average query response time

VI. Conclusion and Future Work

In this paper, we carried out a set of real experiments using our parallel IR system in order to improve the load balance for term partitioning scheme. We proposed to partition the terms of the global index based on term length and the total term frequency extracted from the inverted lists.

We compared our proposed methods with round robin partitioning scheme and the partitioning scheme based on the length of the inverted list. Our results showed that the term length-partitioning scheme performed slightly better than other schemes with respect to node utilization (Table VI). On the other hand, partitioning terms based on the length of the inverted list achieved slightly less average query response time than other schemes (Table VIII).

Acknowledgment

The authors would like to thank Al-Zaytoonah University for their support.

References

[1] Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press, Addison-Wesley, New York (1999)

[2] Zobel, J.,Moffat, A.: Inverted Files for Text Search Engines, ACM Computing Surveys (CSUR) (2006)

[3] Zobel, J.,Moffat, A., Ramamohanarao, k.: Inverted Files Versus Signature Files for Text Indexing, ACM Transactions on Database systems, 453-490 (1998)

[4] Moffat, A.,Webber, W.,Zobel ,J.: Load Balancing for Term-Distributed Parallel Retrieval, The 29th annual international ACM SIGIR conference on Research and development in information, 348-355. ACM, New York (2006)

[5] Xi, W., Somil, O., Luo, M., and Fox, E.: Hybrid partition inverted files for large-scale digital libraries.In Proc. Digital Library: IT Opportunities and Challenges in the New Millennium, Beijing, China, Beijing Library Press (2002)

[6] Cambazoglu, B., Catal, A., Aykanat, C.: Effect of Inverted Index Partitioning Schemes on Performance of Query Processing in Parallel Text Retrieval Systems. A. Levi et al. (Eds.): ISCIS 2006, LNCS, 4263(6), 717–725. Springer, Heidelberg (2006)

[7] Jeong, B.S., Omiecinski, E.: Inverted File Partitioning Schemes in Multiple Disk Systems, IEEE, Transactions on Parallel and Distributed Systems, 6(2), 142-153. IEEE press, Piscataway, NJ, USA (1995)

[8] Heinz, S., Zobel, J.: Efficient Single-Pass Index Construction for Text Databases. Journal of the American Society for Information Science and Technology, 54(8), 713-729 (2003)

[9] Jaruskulchai, C., Kruengkrai,C.: Building Inverted Files Through Efficient Dynamic Hashing (2002)

[10] Badue, C., Baeza-Yates, R., Ribeiro-Neto, B., Ziviani, N.: Distributed Query Processing Using Partitioned Inverted Files, 10-20 (2001)

[11] Lester, N., Moffat, A., Zobel, J.: Fast On-Line Index Construction by Geometric Partitioning, CIKM'05, October 31 and November 5, Proceedings of the 14th ACM international conference on Information and knowledge management , Bremen, Germany, 776-783 ACM (2005)

[12] Case, D. *Looking for Information: A survey of research on Information Seeking, Needs, and Behavior*. USA: Elsevier Science. pp:140-141 (2002)

[13] Abusukhon, A., Talib, M. and Oakes, M.P. Improving the Load Balance for Hybrid Partitioning Scheme by Directing Hybrid Queries. In: Burkhart, H. (Eds.). P*roceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks as part of the 26th IASTED International Multi-Conference on APPLIED INFORMATICS.* Innsbruck, Austria 12-14 February 2008, pp. 238-244. ACTA press: USA. (2008a).

[14] Abusukhon, A. and Oakes, M.P. An Investigation into Query Throughput and Load Balance Using Grid IR. In *Proceedings of the 2nd BCS- IRSG Symposium on Future Directions in Information Access FDIA 2008*. BCS London Office, UK, 22nd September 2008, pp. 38-44. eWic: UK. (2008b)

[15] Abusukhon, A., Oakes, M. Talib, M. and Abdalla, A. Comparison Between Document-based, Termbased and Hybrid Partitioning. In Snasel,V. et al. (Eds.), *Proceedings of the First IEEE InternationalConference on the Application of Digital Information and Web Technologies.* Ostrava, Czech Republic, 4-6 August, pp. 90-95. IEEE, (2008c)

[16] Abusukhon, A. and Talib, M. Improving Load Balance and Query Throughput of Distributed IR Systems. International Journal of Computing and ICT Research (IJCIR), 4(1), pp 20-29, (2010)

[17] Gulli, A. and Signorini, A., The indexable web is more than 11.5 billion pages. The 14th international conference on World Wide Web ACM, New York, USA, pp 902-903, (2005).

[18] Sullivan,D., Searches per day. Search Engine. Watch,http://searchenginewatch.com/reports/article.php/2156461,(2003)

[19] Marin, M., and Costa, G.V. High-Performance Distributed Inverted Files. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management* CIKM'07. Lisbon, Portugal, 6-9 November 2007, pp. 935-938, ACM: New York, USA, (2007).

[20] Jeong, B.S., and Omiecinski, E. Inverted File Partitioning Schemes in Multiple Disk Systems. *IEEE Transactions on Parallel and Distributed Systems*, 6(2), pp. 142-153. IEEE Press, USA. (1995).

Authors Profile

Dr. Ahmad Abusukhon got his Bachelor degree in Computer Science from Mu'tah University in 1990, his M.Sc degree in Computer Science from the University of Jordan in 2001 and he got his PhD degree in Computer Science from the University of Sunderland in 2009. He is now working as assistant Prof. at Al-Zaytonnah University. Dr. Abusukhon is intersted in Computer networks, Distributed systems, and Distributed computing.

Professor M. Talib has, presently, been associated with the computer science department of the university of Botswana and has also been an adjunct professor at the Touro University International (TUI), USA. He has worked at a number of universities all across the globe in different capacities besides India where he remained the Head of the Department of Computer Scienc. He has an excellent industrial relevance and has worked as Software Engineer at the silicon valley in California for a significant period of time. He has been a Consultant for several software development companies and handled various small and big projects all across the world. He was conferred upon a degree of the Doctor of Philosophy (Ph.D.) in computer science with specialization in computer vision from the prestigious University of Lucknow in India with Certificate of Honor. Besides PhD, he is also flanked by an M.S. in computer science, MSc in statistics and PG Diploma in Computing. He has supervised over a dozen Master and four PhD students in different areas of Computer Science, Business and IT. His research areas include Bio informatics, Computer Vision, and Robotics. Presently, he is working on a two way interactive video communication through the virtual screen with the essence of smell. He has over sixty five research papers published in different world class journals and conferences besides a book. He is also credited with over 300 publications including (under)graduate project reports, thesis, extension articles, study guides, edited research papers, books, etc. besides a minimum of 50 Industrial training supervision reports all across the world. He has chaired and remained member of various Academic Councils, Board of Studies, Academic and Advisory Boards, Examination Committees, Moderation and Evaluation Committees worldwide. He is the Member of the Editorial Board of about a dozen International Journals. He has also been associated with a number of international computer societies, associations, forums etc. in various capacities.